

뉴메레르(Numeraire)

해당 번역본은 이용자가 디지털자산 관련 정보를 편리하게 확인할 수 있도록
단순 참고용으로 번역된 것이며, 투자 권유를 목적으로 하지 않습니다.
그 내용이 정확하지 않을 수 있으므로 정확한 정보 습득을 위해서는
원문을 참고하시거나 원문 작성 주체 측에 문의하시기 바랍니다.
두나무는 해당 번역본 내용의 진실성 및 정확성을 보장하지 않고,
제공된 정보에 의한 투자 결과에 대하여 어떠한 책임도 지지 않으며,
투자 손실은 투자자에게 귀속됩니다

Numerai 토너먼트

개요

소개

Numerai는 주식시장을 예측하기 위한 머신러닝 모델을 구축하는 데이터 과학 대회다. 참가자는 모델 학습에 사용할 수 있는 고품질의 무료 데이터를 제공받으며, 매일 예측 결과를 제출할 수 있다. Numerai는 이후 한 달간 이 예측의 성과를 계산하고, 사용자는 자신의 모델에 NMR을 스테이킹하여 성과에 따라 보상을 받거나 일부가 소각된다.

[회원가입](#)을 통해 참여하고, [홈페이지](#)에서 다양한 튜토리얼을 확인할 수 있다.

데이터

Numerai의 무료 데이터셋은 정제되고 규칙화된 금융 데이터로 구성되어 있다. 데이터셋은 **비식별화 (obfuscated)** 되어 있어 무료로 배포할 수 있으며, 금융 도메인 지식 없이도 모델링이 가능하다. 그러나 이러한 비식별화로 인해 이 데이터로 구축된 모델은 Numerai 토너먼트 외부에서 사용할 수 없다.

다음은 데이터셋의 일반적인 구조 예시다:

id	era	feature1	...	feature310	target
n2b2e3dd163cb422	era1	0.75	...	0.00	0.25
n177021a571c94c8	era1	1.00	...	0.25	0.75
n7830fa4c0cd8466	era1	0.25	...	1.00	0.00
nc584a184cee941b	era1	0.25	...	0.00	1.00
nc5ab8667901946a	era1	0.75	...	0.25	0.25
n84e624e4714a7ca	era1	0.00	...	0.75	1.00

데이터셋의 각 행(row)은 특정 시점의 특정 주식을 나타낸다. 시점은 **era**로 표시되며, 각 **era**는 일주일의 의미를 가진다. 각 **era** 내의 ID는 고유하며, 이는 서로 다른 **era** 간에 동일한 주식을 연결할 수 없도록 하기 위한 비식별화 과정의 일환이다. **features**는 해당 시점에서 주식에 대해 알려진 정량적 속성(예: P/E 비율, ADV 등)을 나타내며, **target**은 20일 후의 주식시장 수익률을 측정한 값으로, 낮을수록 부진한 성과를, 높을수록 우수한 성과를 의미한다.

다음은 데이터셋을 불러오는 예시이다:

```
from numerapi import NumerAPI
import pandas as pd

napi = NumerAPI()
napi.download_dataset("v5.0/train.parquet")
training_data = pd.read_parquet("v5.0/train.parquet")
```

자세한 내용은 [Data](#) 섹션을 참고하라.

모델링

참가자의 목표는 주어진 **features**를 기반으로 **target**을 예측하는 머신러닝 모델을 구축하는 것이다. 원하는 언어나 프레임워크를 자유롭게 사용할 수 있다.

다음은 [LightGBM](#)을 사용한 Python 예시이다:

```
import lightgbm as lgb
```

```
features = [f for f in training_data.columns if "feature" in f]

model = lgb.LGBMRegressor(
    n_estimators=2000,
    learning_rate=0.01,
    max_depth=5,
    num_leaves=2 ** 5,
    colsample_bytree=0.1
)
model.fit(
    training_data[features],
    training_data["target"]
)
```

더 많은 예시는 [Models](#) 섹션을 참고하라.

제출

매주 화요일부터 토요일까지 매일 새로운 live data가 공개된다. 이는 현재 주식시장의 상태를 반영한 데이터다. 참가자는 live predictions을 생성하여 Numerai에 제출해야 한다. 제출 시에는 live 데이터의 각 id에 대한 예측값을 제공해야 한다.

다음은 제출 파일의 예시다:

id

prediction

n60dffdaceb7e467

0.25

nadaeef0214b84a8

1.00

nb13883520a4344f

0.25

n423766c5a4fa42a

0.75

n252b14301e46a31

0.25

n75a5baf93a624cc

0.00

n2ff91086716e413

1.00

제출 예시

다음은 Python을 사용하여 실시간 예측을 생성하고 업로드하는 방법의 예시다:

```

# Use API keys to authenticate
napi = NumerAPI("[your api public id]", "[your api secret key]")

# Download latest live features
napi.download_dataset(f"v5.0/live.parquet")
live_data = pd.read_parquet(f"v5.0/live.parquet")
features = [f for f in live_data.columns if "feature" in f]
live_features = live_data[features]

# Generate live predictions
live_predictions = model.predict(live_features)

# Format and save submission
submission = pd.Series(
    live_predictions, index=live_features.index
).to_frame("prediction")
submission.to_csv(f"submission.csv")

# Upload submission
napi.upload_predictions(f"submission.csv", model_id="your-model-id")

```

백엔드에서는 Numerai가 모든 모델의 예측을 결합하여 *스테이크 가중치(Meta Model)*를 형성하며, 이 모델은 Numerai 헤지펀드의 실제 거래에 활용된다.

더 많은 세부 내용과 예시는 [Submissions](#) 섹션을 참고하라.

채점

제출된 예측은 두 가지 주요 지표로 평가된다:

- [Correlation](#) (CORR): 예측과 타겟 간의 상관관계
- [Meta Model Contribution](#) (MMC): 메타 모델에 대한 예측의 기여도

target이 20영업일간의 주식시장 수익률을 나타내기 때문에, 각 제출이 완전히 평가되기까지 약 한 달이 소요된다.

자세한 내용은 [Scoring](#) 섹션을 참고하라.

스테이킹

모델의 성과에 자신이 있다면, Numerai의 암호화폐인 [NMR](#)을 스테이킹할 수 있다. 각 제출에 대해 20일 간의 평가가 완료되면, 양의 점수를 얻은 모델은 추가 NMR로 보상받으며, 음의 점수를 얻은 모델은 스테이킹된 NMR 일부가 소각(*burned*) 된다 (이는 완전히 파괴되어 누구도 접근할 수 없다).

스테이킹은 두 가지 중요한 기능을 수행한다:

1. “Skin in the game” 원리에 따라 Numerai는 스테이킹된 예측의 품질을 신뢰할 수 있다.
2. 보상과 소각은 메타 모델의 가중치를 지속적으로 개선한다.

자세한 내용은 [Staking](#) 섹션을 참고하라.

FAQ

스테이킹은 필수인가?

아니다. 스테이킹 없이도 예측 파일을 제출하고 성과를 확인할 수 있다.

다른 모델에 스테이킹할 수 있나?

아니오. 그러나 다른 모델을 다운로드할 수 있는 두 가지 방법이 있다:

1. Numerai는 [벤치마크 모델](#)을 무료로 배포한다.
2. 커뮤니티가 개발한 [모델 거래 마켓플레이스](#)가 있다.

직접 거래에 사용할 수는 없나?

없다. Numerai 토너먼트의 예측은 개인 거래에 사용할 수 없다. 데이터가 비식별화되어 있기 때문에 자체 거래에 활용하는 것은 불가능하다.

왜 USD로 보상하지 않나?

USD는 소각이 불가능하다. NMR은 소각을 염두에 두고 설계되었으며, 무효 주소(null address)로 전송되어 누구도 사용할 수 없게 된다. 이는 중요한데, 성과가 부진할 경우 소각된 NMR이 단순히 다른 사용자에게 이동하는 것이 아니라 실제로 사라진다는 점을 보장하기 때문이다.

지원

질문, 지원, 피드백은 [Discord](#)에서 확인하라!

모델

튜토리얼

Numerai 데이터로 모델을 구축하는 방법을 배우는 가장 좋은 방법은 튜토리얼을 통해 학습하는 것이다:

1. [Hello Numerai](#)
2. [Feature Neutralization](#)
3. [Target Ensembles](#)

벤치마크 모델

Numerai 벤치마크 모델은 Numerai 팀이 직접 구축한 표준 모델 세트다.

이들의 예측값은 매 라운드마다 공개되어, 누구나 손쉽게 이를 제출하거나 스테이킹할 수 있다.

이 모델들은 현재의 최고 수준(state-of-the-art) 모델과 자신의 모델을 비교할 수 있는 좋은 기준점이다.

모델 목록과 최근 성과는 다음에서 확인할 수 있다: numer.ai/~benchmark_models

다운로드

검증(validation) 및 실시간(live) 예측 결과는 [API](#)를 통해 제공된다.

```
from numerapi import NumerAPI
napi = NumerAPI()
napi.download_dataset("v5.0/train_benchmark_models.parquet",
"train_benchmark_models.parquet")
napi.download_dataset("v5.0/validation_benchmark_models.parquet",
"validation_benchmark_models.parquet")
napi.download_dataset("v5.0/live_benchmark_models.parquet",
"live_benchmark_models.parquet")
```

모델 생성 방식

워크 포워드 교차 검증 (Walk Forward Cross Validation)

모든 예측은 워크 포워드(Walk-Forward) 프레임워크를 사용해 생성된다.

이는 모든 예측이 해당 시점 이전에 존재했던 데이터로만 학습된 모델에 의해 수행된다는 것을 의미한다.

구체적으로, 데이터는 156개의 era 단위로 구분된다.

각 era 청크(chunk)에 대한 예측은 first_era_of_chunk - purge_eras까지의 데이터로 학습된 모델에 의해 생성된다.

purge_eras의 개수는 20D 타겟의 경우 8, 60D 타겟의 경우 16이다.

즉, 모델은 era 1부터 148까지로 학습하고, era 149~156은 제외(purge)한 뒤 era 157~312에 대해 예측한다.

그 다음에는 era 1~304로 학습하고, era 305~312를 제외한 후 era 313~468에 대해 예측한다.

이러한 식으로 반복된다.

워크 포워드 검증 윈도우는 아래와 같은 형태가 된다:

윈도우	학습 시작	학습 종료	검증 시작	검증 종료
1	1	148	157	312
2	1	304	313	468
3	1	460	469	624
4	1	616	625	780
...	...	...	...	...

표준 대형 LGBM 하이퍼파라미터

대부분의 모델은 다음과 같은 LGBM 하이퍼파라미터를 사용한다:

```
standard_large_lgbm_params = {
    "n_estimators": 20000,
    "learning_rate": 0.001,
    "max_depth": 6,
    "num_leaves": 2**6,
    "colsample_bytree": 0.1,
}
```

딥 LGBM 파라미터 (Deep LGBM params)

트리의 개수를 늘리는 것이 도움이 되는 경우가 있었으며, 반대로 더 적은 수의 트리에 깊이를 더하는 방식으로 계산 자원을 적게 사용하면서 유사한 성능을 달성할 수 있었다.

이 하이퍼파라미터 연구에 대한 자세한 내용은 [이 포럼 글](#)에서 확인할 수 있다.

v5 데이터가 공개된 이후, 우리는 v5 벤치마크 모델을 학습할 때 사용한 고성능의 “딥(deep)” 파라미터 구성을 발표했다:

```
deep_lgbm_params = {
    "n_estimators": 30000,
    "learning_rate": 0.001,
    "max_depth": 10,
    "num_leaves": 1024,
    "colsample_bytree": 0.1,
    "min_data_in_leaf": 10000
}
```

앙상블 (Ensembles)

모든 앙상블은 다음 단계를 따른다:

1. 각 예측값을 era별로 가우시안화(gaussianize)한다.
2. 표준편차 1로 표준화(standardize)한다.
3. 각 모델에 대한 가중치를 나타내는 가중치 벡터와 예측값을 내적(dot-product)한다.
4. 결과 벡터를 다시 가우시안화한다.
5. (해당되는 경우) 결과 벡터를 중립화(neutralize)한다.

1단계부터 4단계까지의 과정은 다음과 같다:

```
def rank_gauss_pow1(s: pd.Series) -> pd.Series:
    # do rank-normalize
    s_rank = rank_keep_ties_keep_na(s)

    # gaussianize
    s_rank_norm = pd.Series(stats.norm.ppf(s_rank), index=s_rank.index)

    # Standardize to 1 std
    result_series = s_rank_norm / s_rank_norm.std()

    return result_series

ensemble_cols = ["V4_LGBM_NOMI20", "V42_RAIN_ENSEMBLE"]
weight_vector = [0.1, 0.9]
for col in X[ensemble_cols]:
    if "era" in X.columns:
        X[col] = X.groupby("era", group_keys=False)[col].transform(lambda
s1: rank_gauss_pow1(s1))
    else:
        # check X contains only a single era
        assert 1800 < X.shape[0] < 6000
        X[col] = rank_gauss_pow1(X[col])
return X[ensemble_cols].dot(weight_vector)
```

중립화 (Neutralization)

일부 모델은 중립화(neutralization) 과정을 포함하고 있다.

이 과정은 기본적으로 회귀(regression)를 수행하여 예측값이 각 피처(feature)에 얼마나 노출되어 있는지를 계산한 다음, 그 노출도를 예측 벡터에서 제거함으로써 결과적으로 모든 피처에 대해 직교(orthogonal)한 벡터를 만드는 것이다.

다음은 특정 피처 목록(neutralizers)을 기준으로 여러 벡터(열)를 중립화하는 코드 예시다:

```

def neutralize(
    df, columns, neutralizers=None, proportion=1.0, era_col="era"
):
    if neutralizers is None:
        neutralizers = []
    unique_eras = df[era_col].unique()
    computed = []
    for u in unique_eras:
        df_era = df[df[era_col] == u]
        scores = df_era[columns].values
        scores2 = []
        for x in scores.T:
            x = pd.Series(x)
            x = (x.rank(method="first") - 0.5) / len(x.dropna())
            x = stats.norm.ppf(x)
            scores2.append(x)
        scores = np.array(scores2).T
        exposures = (
            df_era[neutralizers]
            .fillna(df_era[neutralizers].median())
            .fillna(0.5)
            .values
        )

        scores -= proportion * exposures.dot(
            np.linalg.pinv(exposures.astype(np.float32), rcond=1e-6).dot(
                scores.astype(np.float32)
            )
        )

        scores /= pd.DataFrame(scores).std(ddof=0, axis=0,
skipna=True).values

        computed.append(scores)

    return pd.DataFrame(np.concatenate(computed), columns=columns,
index=df.index)

```

이 모델들은 무엇인가?

많은 벤치마크 모델은 다음과 같은 명명 규칙을 따른다:

`{data_version}_LGBM_{target}`

즉, 데이터 버전(V2, V3, V4, V41, V42, V43, V5 등)과 타깃(e.g. cyrusd_20, teager2b_20 등)을 조합한 이름을 사용한다.
이들은 지정된 데이터 버전과 타깃을 기반으로, 표준 워크 포워드 방식과 표준 LGBM 파라미터로 학습된 모델들이다.
그게 전부다!

이 명명 규칙을 따르지 않는 고유한 모델들도 존재한다:

V5_LGBM_CT_BLEND (곧 공개 예정)

이는 V5_LGBM_TEAGER2B20과 V5_LGBM_CYRUSD20을 50:50 비율로 블렌딩한 단순 혼합 모델이다.

다음 모델들은 벤치마크 모델 페이지에 표시되어 있지만, 예측 파일에는 포함되어 있지 않다.
이는 재현이 매우 용이하기 때문이다:

INTEGRATION_TEST – 당시 가장 우수하다고 판단된 모델을 제출한다. V2, V3, V4 예측을 거쳐 현재는 v5_lgbm_ct_blend로 전환되었다.

NB_HELLO_NUMERAI – 기본 “Hello Numerai” 튜토리얼 노트북에서 생성된 모델을 제출한다.

NB_FEATURE_NEUTRAL – “Feature Neutralization” 튜토리얼 노트북에서 생성된 모델을 제출한다.

NB_TARGET_ENSEMBLE – “Target Ensemble” 튜토리얼 노트북에서 생성된 모델을 제출한다.

NB_EXAMPLE_MODEL – “Example Model” 노트북에서 생성된 기본 예시 모델을 제출한다.

커뮤니티 모델 (Community Models)

Numerai 커뮤니티는 [Numerbay](#)라는 플랫폼을 개발했다.

이는 Numerai 커뮤니티 구성원들에 의해 만들어진 모델을 사고파는 웹사이트다.

단, Numerai는 Numerbay에 등록된 모델의 성능을 보장하지 않는다.

제출 (Submissions)

제출이란 무엇인가?

제출은 순위화된 수익률(returns)을 예측하려는 실수(float) 벡터다.

예측값은 0과 1 사이여야 하며,

0은 가장 낮은 수익률, 0.5는 평균 수익률, 1은 가장 높은 수익률을 의미한다.

우리의 데이터에 포함된 타깃은 모델이 이러한 수익률을 예측하도록 학습되도록 설계되어 있다.

모든 제출은 Numerai가 실제 거래에 사용하는 메타 모델(Meta Model)에 통합된다.

라운드 (Rounds)

각 제출은 토너먼트의 한 “라운드(Round)”에 속한다.

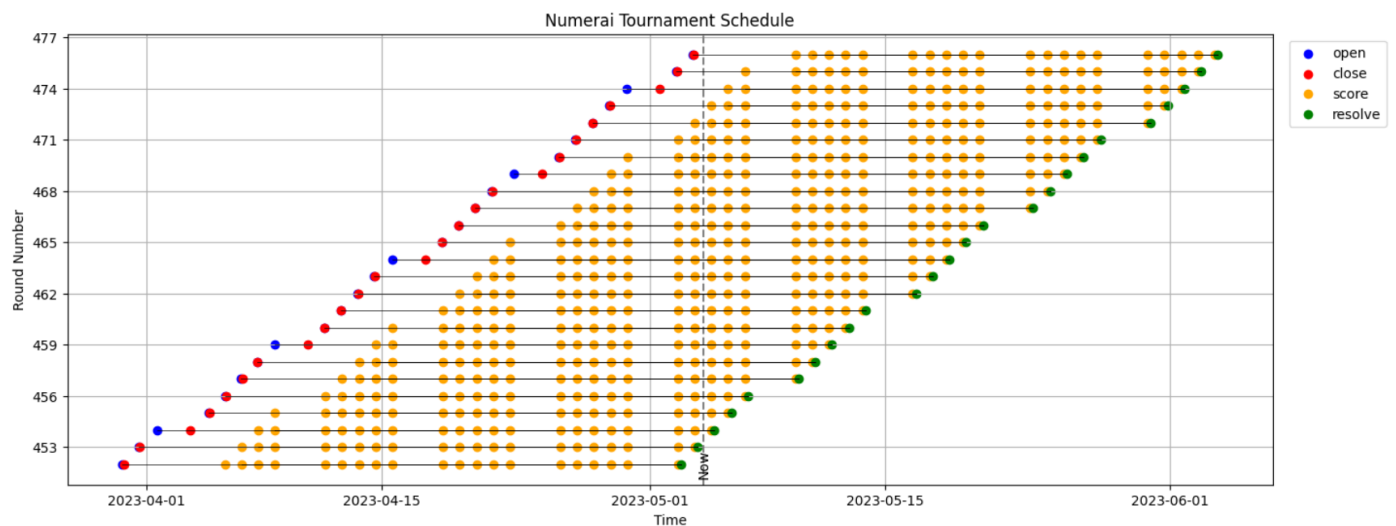
각 라운드는 약 한 달에 걸쳐 4단계를 거친다:

- **Open:** 새로운 라운드가 시작되고, 새로운 라이브 피처가 공개된다.
- **Close:** 라이브 예측을 제출해야 하는 마감 시점이다.
- **Score:** 제출된 예측이 평가되는 기간이다.
- **Resolve:** 최종 점수 및 보상이 확정되는 시점이다.

새로운 라운드는 매주 화요일부터 토요일까지 매일 시작된다.

각 라운드는 약 한 달간 진행되므로, 매주 5개의 라운드가 시작되어 서로 겹치는 일정이 형성된다.
즉, 동시에 최대 **25개의 라운드**가 중첩되어 진행될 수 있다.

다음은 라운드 일정의 시각적 예시다:



라운드 453-477의 중첩 시각화

토요일에 시작하는 라운드는 주말 전체를 차지하며, 토요일에 열리고 월요일에 마감된다.

반면, 화요일~금요일에 시작하는 라운드는 **1시간 동안만** 열려 있다.

또한 일요일과 월요일에는 점수가 공개되지 않는다.

일반적인 라운드 진행 일정은 다음과 같다:

라운드	오픈 시간*	마감 시간*	점수 시작	최종 확정
화요일	Tue 13:00 UTC	Tue 14:00 UTC	다음 토요일	약 31일 후 (금요일)
수요일	Wed 13:00 UTC	Wed 14:00 UTC	다음 화요일	약 31일 후 (토요일)
목요일	Thu 13:00 UTC	Thu 14:00 UTC	다음 수요일	약 33일 후 (화요일)
금요일	Fri 13:00 UTC	Fri 14:00 UTC	다음 목요일	약 33일 후 (수요일)
토요일	Sat 13:00 UTC	Mon 14:00 UTC	다음 금요일	약 33일 후 (목요일)

* 실제 오픈 및 마감 시간은 라운드마다 약간 달라질 수 있다.

그러나 모든 라운드는 **최소 1시간의 제출 창(submission window)**을 보장하며,
항상 **13:00 UTC 이후에 열리고 14:00 UTC 이전에 마감되지 않는다.**

제출하기

토너먼트에서 올바르게 경쟁하려면 매 라운드마다 실시간 예측(live predictions)을 제출해야 한다.

다음은 [NumerAPI](#)를 사용해 Python에서 제출을 수행하는 예시다:

```
from numerapi import NumerAPI
import pandas as pd

# 인증
napi = NumerAPI("[your api public id]", "[your api secret key]")

# 현재 라운드 가져오기
current_round = napi.get_current_round()

# 최신 라이브 피쳐 다운로드
napi.download_dataset(f"v5.0/live_{current_round}.parquet")
live_data = pd.read_parquet(f"v5.0/live_{current_round}.parquet")
live_features = live_data[[f for f in live_data.columns if "feature" in f]]

# 실시간 예측 생성
live_predictions = model.predict(live_features)

# 제출 파일 포매팅
```

```
submission = pd.Series(live_predictions,
index=live_features.index).to_frame("prediction")
submission.to_csv(f"prediction_{current_round}.csv")

# 제출 업로드
napi.upload_predictions(f"prediction_{current_round}.csv",
model_id="your-model-id")
```

다중 제출 (Multiple Submissions)

제출 기간(submission window) 동안 원하는 만큼 여러 번 제출할 수 있다.
단, 평가 및 보상에 반영되는 것은 **가장 마지막으로 제출된 유효한 제출물**이다.

지연 제출 (Late Submissions)

현재 라운드의 제출 기간을 놓쳤더라도 여전히 제출할 수 있다.
이 경우 제출은 “지연(late)”으로 간주되며, 점수는 부여되지만 스테이킹은 불가능하다.
지연 제출은 메타 모델에 영향을 주지 않으며, 해당 제출의 위험 노출 NMR(at-risk NMR)은 0이다.
또한 다른 사용자의 보상 계수(payout factor)에도 영향을 미치지 않는다.

자동화 (Automation)

모델 업로드 (Model Uploads)

가장 간단하면서도 제약이 많은 자동화 방법은 학습된 모델을 Numerai에 업로드하는 것이다.
한 번 업로드하면 Numerai가 매일 자동으로 예측을 생성하고 제출을 처리한다.
인프라나 서버 관리가 번거로운 경우 이 방법을 사용하는 것이 좋다.

자세한 내용은 [model upload](#) 섹션을 참고하라.

Numerai-CLI

“Compute Heavy”라고도 불리는 공식 셀프 호스팅 클라우드 솔루션이다.
AWS, GCP, Azure 환경을 모두 지원하며, 복잡한 모델이나 파이프라인을 운영하거나 직접 자동화를 구축하려는 경우 권장된다.
Numerai는 이 프레임워크를 적극적으로 지원하며, 버그 리포트 및 기술 지원 요청에 직접 대응한다.
또한 높은 안정성과 저비용을 목표로 설계되어 있다.

자세한 내용은 [Numerai-CLI GitHub 페이지](#)를 참고하라.

로컬 서버 (Local Server)

온프레미스 환경에서 직접 운영하고 싶다면 로컬 서버 방식을 사용할 수 있다.
상대적으로 단순하고 비용이 저렴하지만, 신뢰성을 유지하고 문제를 직접 해결해야 한다는 부담이 있다.

- [CRON 예시](#) – 스크립트를 정기적으로 실행하고 제출 시점을 자동 감지하도록 설정하는 방법
- [NGROK 예시](#) – Numerai가 제출 알림을 보낼 수 있도록 웹훅(webhook)을 설정하는 방법

대기 및 지연 제출 (Queued and Delayed Submissions)

현재 라운드의 제출 기간을 놓친 경우, 제출물은 자동으로 다음 라운드의 대기열(queued)에 추가된다.

대기 제출은 다음 라운드가 시작되면 즉시 “정시 제출(on-time submission)”로 전환된다.

매 라운드마다 live data의 ID가 변경되므로, Numerai가 이전 라운드의 예측 ID를 최신 라운드의 ID로 자동 매핑한다.

만약 파이프라인 실행 시간이 길어져 (예: 24시간 이상 소요될 경우) 제출이 지연된다면,

이전 라운드에서 생성한 예측 결과를 현재 라운드에 제출할 수도 있다.

이 경우에도 Numerai는 이전 라운드의 예측 ID를 자동으로 최신 라운드의 ID로 매핑한다.

모델 업로드 (Model Uploads)

소개

모델 업로드는 일일 제출을 자동화하는 가장 간단하고 무료인 방법이다.

학습된 모델을 .pkl 파일 형태로 업로드하면 Numerai가 자동으로 예측을 생성하고 제출을 처리한다.

따라서 안정성, 인프라, 스케줄링 등에 신경 쓸 필요가 없으며,

사용자는 데이터 사이언스에만 집중하면 된다.

자세한 튜토리얼은 [이 노트북](#)에서,

기본 예시는 [이 노트북](#)에서 확인할 수 있다.

작동 방식

1. 학습된 모델 (Trained Model)

[Numerai Predict](#) 실행 환경의 요구 사항을 설치하고 모델을 학습시킨다.

이 환경은 모델이 항상 정상적으로 실행되도록 보장하기 위해 엄격한 패키지 버전 제약을 가진다.

목록에 있는 패키지의 마이너 버전이나 패치 버전을 사용할 수 있지만, 반드시 해당 환경과의 하위 호환성을 유지해야 한다.

목록에 없는 패키지는 사용할 수 없지만, 필요 시 [Discord](#)를 통해 요청할 수 있다.

2. 예측 파이프라인 (Prediction Pipeline)

모델을 함수로 감싸, 라이브 피쳐(live features)를 입력받아 실시간 예측값을 반환하도록 구성한다:

```
# Wrap your model with a function that takes live features and returns
live predictions
def predict(
    live_features: pd.DataFrame,
    live_benchmark_models: pd.DataFrame
) -> pd.DataFrame:
    live_predictions = model.predict(live_features[feature_cols])
    submission = pd.Series(live_predictions, index=live_features.index)
    return submission.to_frame("prediction")
```

이 방식만이 Numerai가 사용자를 대신해 모델을 올바르게 실행할 수 있는 유일한 방법이다. 만약 업로드한 pickle 파일에 아래와 같은 형태의 predict 함수 시그니처가 없다면, 모델 실행은 실패하게 된다.

3. Cloudpickle

[cloudpickle](#) 라이브러리를 사용해 함수를 피클링(pickling)하고, 생성된 pickle 파일을 Numerai에 업로드한다:

```
# Use the cloudpickle library to serialize your function
import cloudpickle
p = cloudpickle.dumps(predict)
with open("predict.pkl", "wb") as f:
    f.write(p)
```

이 라이브러리는 [Dask](#)나 [Ray](#) 같은 분산 컴퓨팅 프레임워크에서도 사용되며, 로컬에서 작성한 Python 코드를 손쉽게 클라우드 환경에서 원격으로 실행할 수 있게 해준다.

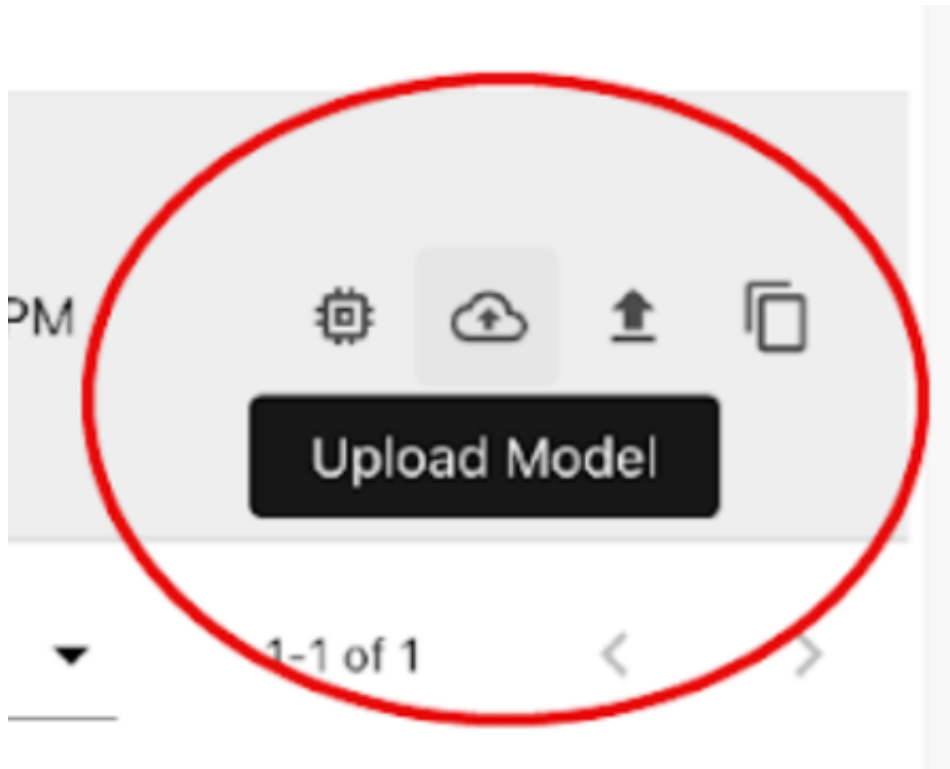
[cloudpickle](#)을 사용하는 주요 이점은 표준 [pickle](#) 라이브러리와 달리 코드뿐만 아니라 **로컬 실행 컨텍스트(local context)**까지 함께 직렬화한다는 점이다.

이 덕분에 로컬 환경에서 개발된 코드를 별도의 추가 설정 없이도 손쉽게 패키징할 수 있다.

위 예시에서 predict 함수는 전역 스코프에 정의된 model과 feature_cols를 참조한다. [cloudpickle](#)은 이 두 객체(model과 feature_cols)를 값으로 직렬화하여, Numerai가 해당 함수를 실행할 때 동일한 객체를 사용할 수 있도록 보장한다.

4. 피클 업로드 (Upload the Pickle)

[Submissions 페이지](#)의 오른쪽에 있는 **Upload Model** 버튼을 클릭하여 모델을 업로드한다:



피클 파일을 선택하고, 사용한 Python 버전을 지정한 다음 **Upload**를 클릭한다.

×

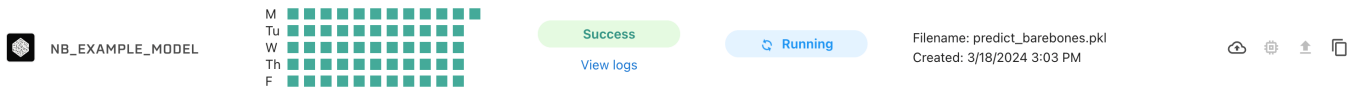
AUTOFOX

[Documentation](#)

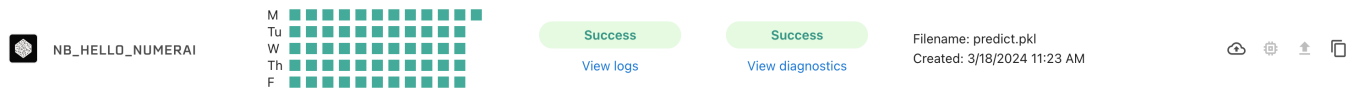
Upload

[illegible][View logs](#) Pending

성공적으로 실행되면 “submissions” 옆에 새로운 블록이 나타나며, 모델의 실행 로그를 검토할 수 있고, Numerai가 검증(validation) 데이터셋에 대해 자동으로 진단(diagnostics)을 수행한다:



진단이 완료되면, 해당 모델에 대한 상세 진단 결과를 볼 수 있다:



5. 일일 실행 (Running Daily)

모델이 업로드되면 Numerai가 매일 모델을 실행해 실시간 예측을 생성하고 자동으로 제출한다. 모든 과정이 정상적으로 동작하면, 최신 제출 상태가 다음 4단계를 순환한다:

1. **Pending:** Numerai가 모델 실행을 위한 클라우드 리소스를 할당 중이다.
2. **Running:** Numerai가 현재 모델을 실행 중이다.
3. **Validating:** 모델 실행이 완료되어 제출 결과를 검증 중이다.
4. **Success:** Numerai가 제출을 성공적으로 수락했다.

문제가 발생할 경우 두 가지 상태 중 하나가 표시된다:

1. **Error:** Numerai가 모델 실행 중 예상치 못한 오류를 발견했다. 팀이 문제를 조사한다.
2. **Failed:** 모델 실행에 실패했다. 로그를 확인하고 정상 작동하는 모델을 다시 업로드해야 한다.
 - Python 또는 의존성 버전 불일치
 - 잘못된 제출 포맷
 - 메모리 부족(Out of memory)
 - 실행 시간 초과(Timeout) 등의 이유가 포함된다.

6. 피클 비활성화 (Disabling the Pickle)

업로드된 모델을 비활성화하고 싶다면 다음 단계를 따른다:

1. [Submissions 페이지](#)로 이동한다.
2. **Model Upload** 버튼을 클릭한다.

NUMERAI

1 NMR = \$17.24 Tournaments Hedge Fund Discord GitHub Forum Docs

Home Leaderboard Season Data Models Submissions Scores Staking

Submissions

Current Round: 843

Models (57) Round 843 models submitted: 10 / 57

NAME	SUBMISSIONS	LATEST SUBMISSION	DIAGNOSTICS	AUTOMATION INFO
INTEGRATION_TEST	M Tu W Th F	—	—	—
NB_HELLO_NUMERAI	M Tu W Th F	Success View logs	Success View diagnostics	Filename: predict.pkl Created: 9/27/2024 11:25 AM

3. “Settings” 탭에서 **Disable**을 선택한다.

MODEL UPLOAD

Model Name NB_HELLO_NUMERAI

Filename predict.pkl

Dataset Version v5.0

Python Version Python 3.10

Uploaded 9/27/2024 11:25 AM

New Upload Settings

Disable model and clear configuration

Disable

FAQ

어떤 Python/패키지 버전이 지원되나?

기본 버전은 **Python 3.11**이지만, **3.10 이상 3.12 이하**의 버전이 지원된다.
업로드 시 원하는 버전을 선택할 수 있다.

Model Upload 환경에서 사용되는 패키지 버전은 다음 파일에서 확인할 수 있다:

<https://github.com/numerai/numerai-predict/blob/master/requirements.txt>

모델이 인터넷에 접근할 수 있나?

아니다. Numerai는 모델의 인터넷 접근 권한을 허용하지 않는다.

메모리(Mem) 및 CPU 제한은 어떻게 되나?

기본적으로 각 모델에는 **1 CPU / 4GB RAM**의 리소스가 할당되며, 최대 **10분간의 실행 시간**이 주어진다 (대기열 시간은 제외).

LightGBM 벤치마크 기준:

- 작은 피쳐 세트로 20K 트리 모델은 1분 이내에 실행
- 전체 피쳐 세트로 90K 트리 모델은 6분 이내에 실행

여전히 수동으로 예측을 업로드할 수 있나?

아니다. 모델을 업로드한 이후에는 API를 통해 수동 제출을 할 수 없다.

모델에 웹훅(webhook)을 설정할 수 있나?

불가능하다.

모델 업로드 후에는 경합 조건(race condition)을 방지하기 위해 웹훅 설정이 비활성화된다.

모델을 업로드하려면 기존의 Compute 설정을 비활성화해야 한다.

Numerai가 내 학습된 모델에 완전히 접근할 수 있나?

그렇다. Numerai는 업로드된 모델을 언피클(unpickle)하여 접근할 수 있다.

만약 이에 불편함이 있다면, [다른 자동화 솔루션](#) 사용을 고려해야 한다.

서비스 이용 약관 (Terms of Service)

Numerai는 보안, 악용, 계정 비활성화, 성능 저하 등 어떠한 이유로든 모델을 비활성화할 권리를 가진다.

Numerai는 이 기능의 사용을 최대한 지원하지만,

최종적으로는 사용자가 자신의 제출 파이프라인이 올바르게 구성되었는지 책임져야 한다.

모델이 100% 정상적으로 제출된다는 보장은 제공되지 않는다.

또한 Numerai는 업로드된 모델의 성능으로 인해 발생할 수 있는

이익 또는 손실에 대해 어떠한 책임도 지지 않는다.

모델의 성과에 대한 책임은 전적으로 사용자에게 있다.

점수 체계 (Scoring)

점수 (Scores)

현재 보상(payout)에 사용되는 주요 점수는 두 가지다:

- 상관관계(Correlation) (CORR): 예측값과 타겟 간의 상관관계
- 메타 모델 기여도(Meta Model Contribution) (MMC): 예측값이 메타 모델에 기여하는 정도

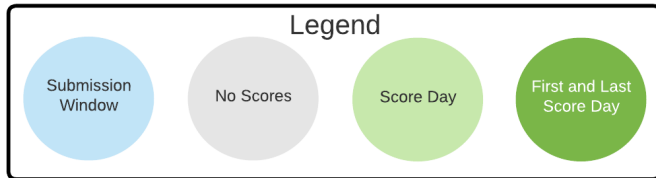
또한, 보상에는 사용되지 않지만 참고용으로 제공되는 정보성 점수들도 있다:

- Feature Neutral Correlation (FNC): 모든 피처에 대해 중립화한 후 타겟과의 상관관계
- **Correlation with the Meta Model (CWMM)**: 메타 모델(스테이크 가중 평균 예측)과의 상관관계
- **Benchmark Model Contribution (BMC)**: 스테이크 가중 벤치마크 모델에 대해 중립화한 후 타겟과의 상관관계

실시간 점수 산정 (Live Scoring)

하나의 라운드 내에서 제출된 예측은 최종 점수가 산출되기 전까지 총 **20번의 점수 업데이트**를 받는다.

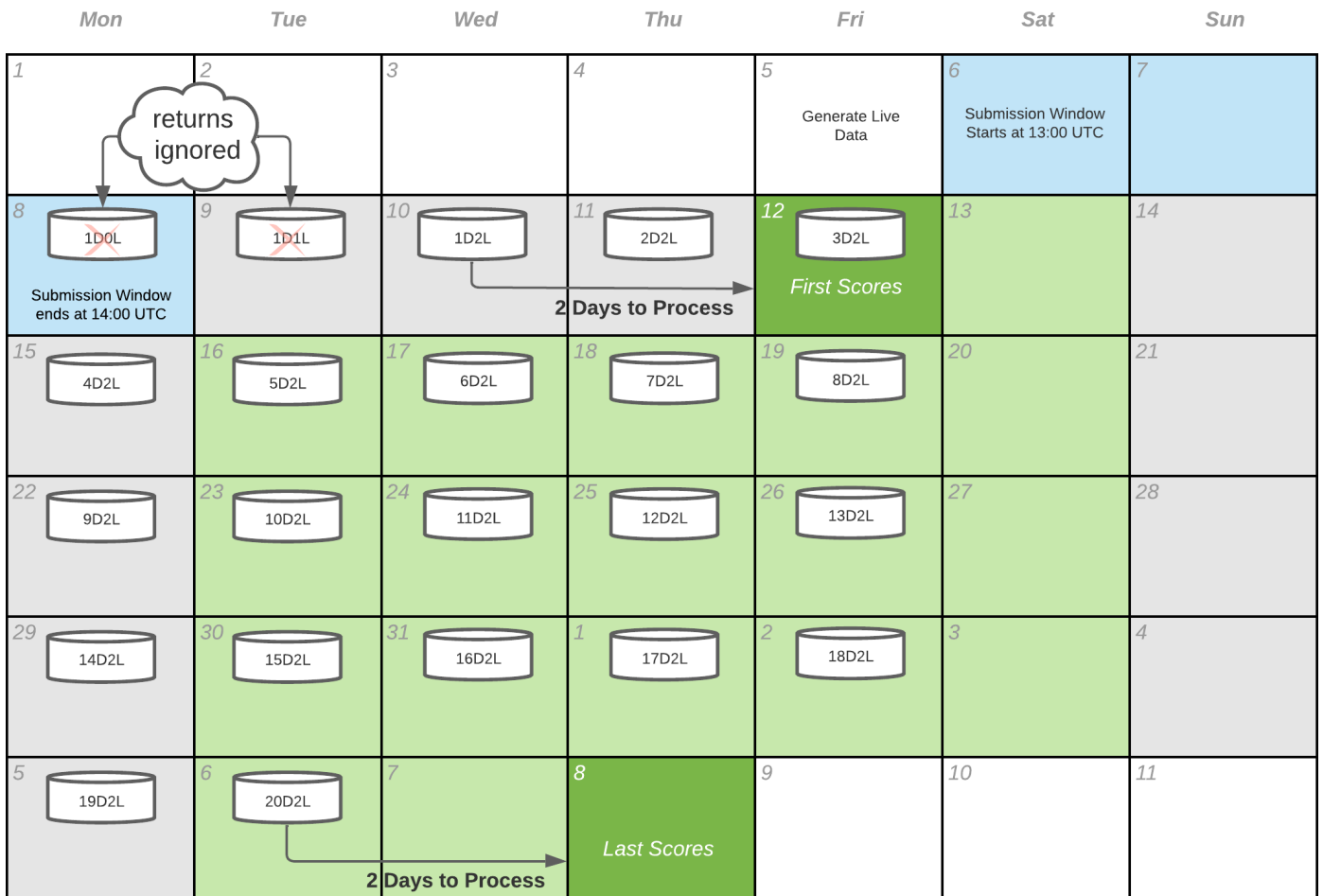
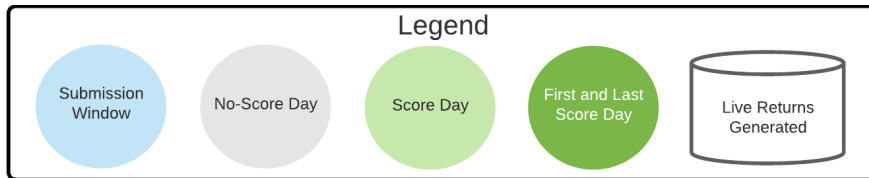
예를 들어, 주말 라운드의 경우를 살펴보자:



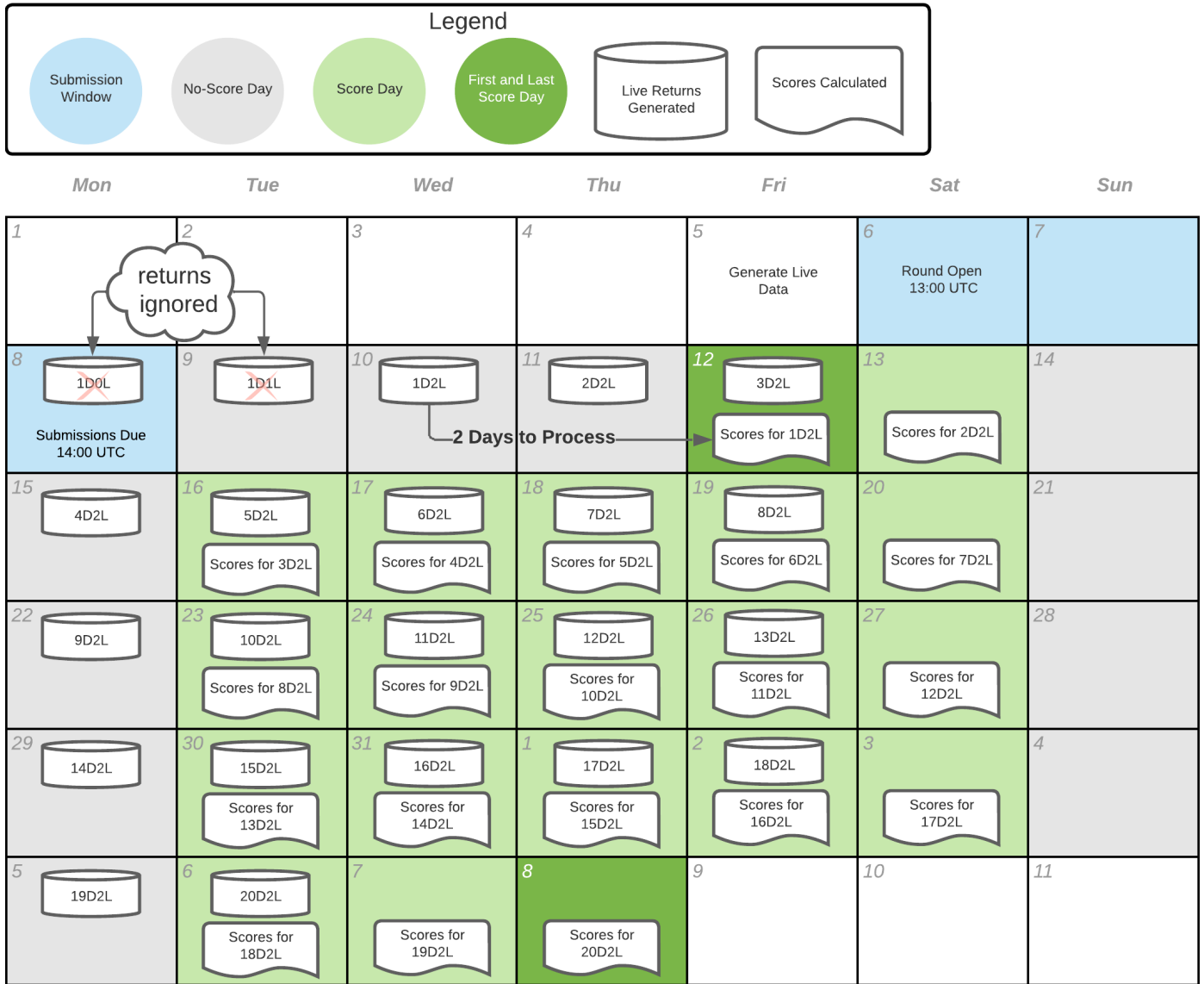
Mon	Tue	Wed	Thu	Fri	Sat	Sun
1	2	3	4	5 Generate Live Data	6 Submission Window Starts at 13:00 UTC	7
8 Submission Window ends at 14:00 UTC	9	10	11	12 First Scores	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29	30	31	1	2	3	4
5	6	7	8 Last Scores	9	10	11

라운드 **토요일 6일에 시작(Open)** 하고, **월요일 8일에 마감(Close)** 된다.
 첫 번째 점수는 **금요일 12일(4일 후)** 에 산출되며, 이후 매주 화요일부터 토요일까지 매일 점수가 갱신된다.
 최종 점수는 약 한 달 후인 **목요일 8일**에 공개된다.

이 점수 산정이 20일 동안 이루어지는 이유는,
 주요 타겟이 **20일간의 누적 수익률(returns)** 을 기반으로 계산되기 때문이다.
 처음 점수가 산출되기까지 4일이 걸리는 이유는,
 데이터 처리에 2일이 필요하고 타겟 계산이 거래일 기준 2일 뒤(lagged)로 지연되기 때문이다.
 즉, 초기 2일의 수익률은 무시하고 20일의 계산 주기를 시작한다.
 이 타임라인은 일반적으로 **"20D2L"**이라 불린다 —
20일의 수익률(Days of returns) + **2일**의 지연(Lag days)을 의미한다.



예를 들어,
 첫 점수가 산출되는 **금요일 12일**은 **1D2L** 타킷을 사용하며,
 이는 **수요일 10일**의 수익률만 포함한다.
 두 번째 점수가 산출되는 **토요일 13일**은 **2D2L** 타킷을 사용하며,
 이는 **수요일 10일 ~ 목요일 11일**의 수익률을 포함한다.
 최종 점수는 약 4주 후, 다음 달 **목요일 8일**에 계산되며,
20D2L 타킷을 기반으로 **수요일 10일 ~ 다음 달 화요일 6일**의 수익률을 포함한다.



또한 Numerai는 **60D2L 점수**도 제공한다.

이는 동일한 로직을 따르지만, 결과가 40 영업일(총 8주 후)에 해석된다.

즉, **총 12주간의 기간**이 소요된다.

60D2L 점수를 기준으로 보상이 지급되는 라운드는 스테이킹된 NMR이 **12주간 잠금(lock-up)** 상태로 유지된다.

리더보드 (Leaderboards)

최종 점수만이 모델의 실시간 성과(live performance)에 반영된다.

순위는 **계정 단위(account level)** 와 **모델 단위(model level)** 모두에서 산출되며, 주요 통합 점수는 reputation이라 부른다.

모델의 reputation은 최근 1년간의 평균 점수이며, 이는 모델 리더보드에서의 순위를 직접적으로 결정한다.

계정의 reputation은 스테이크 가중 평균 점수(Stake-Weighted Average Score) 로 계산된다.
매 라운드마다 각 모델의 점수에 그 모델의 스테이크 비율을 곱해 합산한다.
즉, 스테이크가 없는 단일 모델만 보유하고 있다면,
계정 점수는 해당 모델의 reputation과 동일하다.

Data ScientistsModels

Search

Staked Data Scientists			Staked NMR At Risk				Avg 1Y Return				
475			474,853 NMR				44.2%				
RANK	NAME	CORJ60	FNCV3	CORR20V2	▼ MMC	BMC	CORT20	STAKE	3M	1Y	ALL
1	JOHNDOE45	0.0079	0.0058	0.0196	0.0056	0.0031	0.0037	14,219 NMR	32.2%	107.2%	185.6%
2	EGG	0.0078	0.0061	0.0247	0.0056	0.0029	0.0047	5,478 NMR	25.1%	117.5%	792.5%
3	THEHUMANPEEP	0.0066	0.0038	0.0230	0.0055	0.0013	0.0030	6,796 NMR	6.0%	112.1%	207.0%
4	SUGARCANE	0.0079	0.0070	0.0202	0.0055	0.0047	0.0045	5,413 NMR	56.3%	105.4%	520.4%
5	DERRIDA	0.0093	0.0058	0.0246	0.0054	0.0030	0.0047	4,383 NMR	25.7%	29.0%	131.9%
6	GEORGEMADLIS	0.0036	0.0007	0.0061	0.0052	0.0029	0.0019	12,451 NMR	27.6%	63.6%	12.5%
7	AMBRUL11	0.0007	0.0018	0.0139	0.0052	0.0012	0.0013	37,212 NMR	5.2%	95.0%	88.0%
8	MICKADLR	0.0014	0.0000	0.0099	0.0050	0.0000	0.0000	—	—	63.5%	97.5%
9	BELIXROGNER	0.0002	0.0057	0.0295	0.0046	0.0019	0.0050	4,295 NMR	14.3%	122.8%	469.9%

numer.ai/leaderboard

진단 (Diagnostics)

진단 도구는 검증(validation) 데이터셋을 기반으로 점수를 계산하고 시각화한다.

Performance ↗

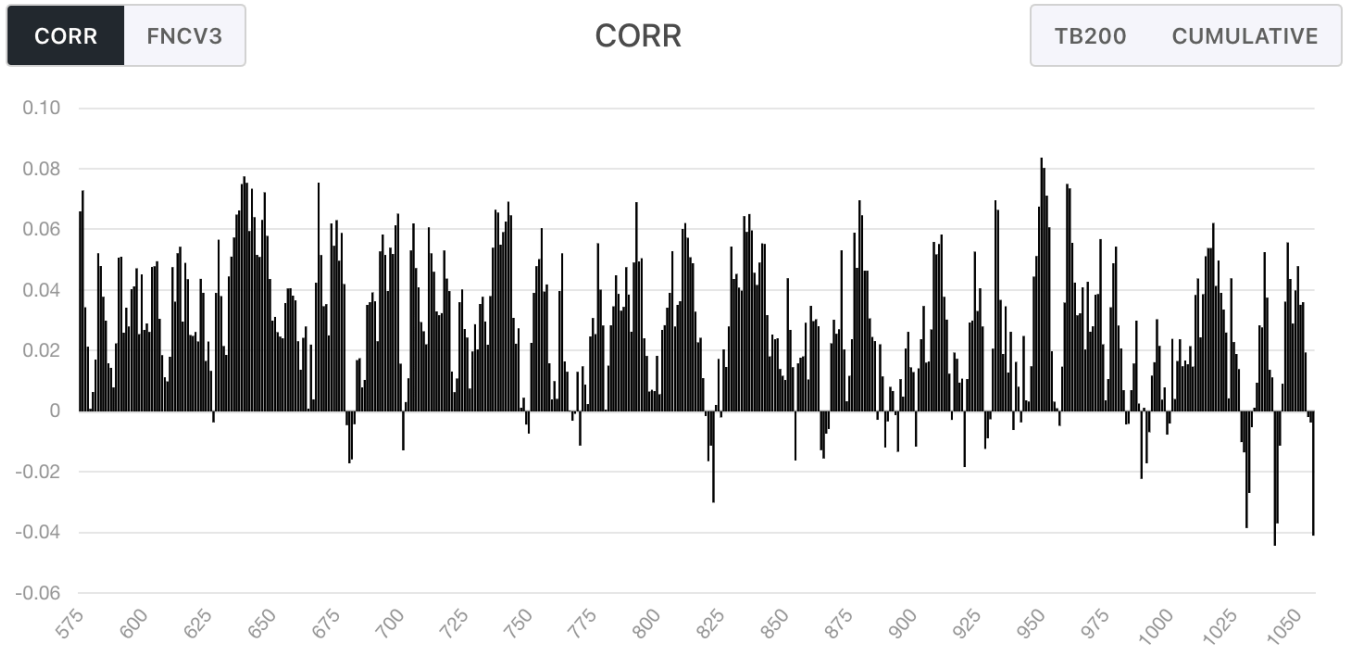
Sharpe Ratio	1.2388
CORR	0.0283
FNCv3	0.0266

Risk ↘

Std. Dev.	0.0228
Feat. Exposure	0.2130
Max Drawdown	-0.0943

Other 🌐

Autocorr	0.0844
FNC	
Ex. Preds Corr	1



진단 보고서 예시

Model Upload를 통해 모델을 업로드한 경우,
Numerai가 자동으로 검증 데이터셋에서 모델을 실행하여 진단 결과를 생성한다.
직접 수행하려면 numer.ai/scores 페이지로 이동해 **Run Diagnostics** 버튼을 클릭하면 된다.

Models (9)										
Search										
NAME	CORRV4	ICV2	RIC	CWSNMM	MCWSM	APCWSM	FNCV4	TC	MMC	STAKE
EXAMPLE_MODEL	0.0019	-0.0151	-0.0010	0.4042	0.4446	0.1569	0.0016	0.0014	0.0009	
INTEGRATION_TEST	0.0019	-0.0151	-0.0010	0.4042	0.4446	0.1569	0.0016	0.0012	0.0009	

⚠️ 주의: 과거의 성과는 미래의 성과를 보장하지 않는다.

특히 금융 머신러닝 분야에서는 검증 지표에 과도하게 의존하면 **오버피팅(overfitting)**의 위험이 있다.

검증 데이터셋에 맞춰 학습한 모델은 실제(out-of-sample) 환경에서 일반화되지 않을 수 있다.

정의 (Definitions)

아래에서는 점수 산출에 사용되는 주요 함수 및 통계 도구의 정의를 설명한다.

오픈소스 코드는 [numeraai-tools/scoring](https://github.com/numeraai/numeraai-tools/scoring)에서 확인할 수 있으며,

다음 명령어로 패키지를 설치할 수 있다:

```
> pip install numeraai_tools
```

통계 (Statistics)

- tie-broken rank
 - [percentile rank](#)을(를) 시리즈에 적용
 - id / index를 기준으로 동률을 분해(break ties)
- tie-kept rank
 - [percentile rank](#)를 시리즈에 적용
 - 동률 집합마다 해당 집합의 tie-broken rank들의 평균을 그 집합의 순위로 설정
- correlation
 - 두 시리즈 간의 상관계수
- spearman correlation
 - 라이브 타깃과 예측 간의 [스피어만 상관계수](#)
 - 스피어만은 동률에 평균 순위를 부여하여 유지하므로 tie-broken-rank correlation과 다르다
- pearson correlation
 - 라이브 타깃과 예측 간의 [피어슨 상관계수](#)
 - 피어슨은 순위를 사용하지 않으므로 다른 상관계수들과 다르다
- tie-broken-rank correlation
 - 라이브 타깃과 tie-broken된 예측 순위 간의 상관관계(정렬된 인덱스, NaN 없음)
 - 참고: 이는 피어슨 상관이지만, 예측을 순위화하므로 동작은 스피어만과 유사하다. 타깃에는 동률이 남고 예측에는 동률이 없기 때문에 1.0의 상관을 달성하는 것은 불가능하다.
- variance normalize
 - 벡터 s를 받아 표준편차를 1로 정규화
- power x / pow x
 - 벡터 s를 받아 부호를 무시하고 각 값을 거듭제곱 x로 승수화

- gaussianize
 - 벡터 s 를 받아 s 의 표준편차로 나누어 단위 노름(unit norm)을 만들기
- neutralization
 - 벡터 s 와 종립화 행렬 N 에 대하여, N 에 대해 직교인 성분 s' 을 찾기:
 - $s' = s - (N \text{ dot } (N_inverse \text{ dot } s))$
- orthogonalize
 - neutralize와 유사하지만, 중심화된 두 열 벡터에 대해 더 빠르다
 - 벡터 u, v 에 대하여, v 의 u 에 대한 직교 성분을 찾기:
 - $v - (u \otimes (\text{dot}(\text{transpose}(v), u) / \text{dot}(\text{transpose}(u), u)))$
- numerai corr
 - 예측 벡터 s 와 타깃 벡터 t 가 주어졌을 때 s 와 t 의 상관을 계산:
 - $s' = \text{tie-kept rank} \rightarrow$ 가우시안화 $\rightarrow$ pow 1.5를 적용한 s
 - $t' = \text{pow 1.5를 적용한 } t$
 - s' 와 t' 의 피어슨 상관을 계산
- feature neutral corr
 - 예측 벡터 s , 종립화할 피쳐 행렬 F , 타깃 벡터 t 가 주어졌을 때, F 에 대해 종립화한 뒤 s 와 t 의 상관을 계산:
 - $s' = \text{tie-kept rank} \rightarrow s$ 를 가우시안화
 - $s'' = s'$ 를 F 에 대해 종립화 $\rightarrow$ 분산 정규화
 - s'' 와 t 의 numerai corr 계산
- correlation contribution
 - 타깃 벡터 t , 메타 모델 벡터 m , 예측 벡터 s 가 주어졌을 때, s 가 m 의 t 에 대한 상관에 기여하는 정도를 계산:
 - $m' = \text{tie-kept rank} \rightarrow$ 가우시안화한 m
 - $s' = \text{tie-kept rank} \rightarrow$ 가우시안화한 s
 - $s'' = s'$ 를 m 에 대해 직교화(orthogonalize)
 - s'' 와 t 의 공분산을 구한다

팩터 & 피쳐 (Factors & Features)

- Factors
 - 데이터 제공업체로부터 받은 암호화되지 않은(정제/포맷 등은 가능) 데이터
 - 사용자에게 제공되지는 않지만 금융에서 매우 잘 알려진 시그널

- 타깃, 포트폴리오, 메타 모델은 항상 이에 대해 중립화한다
- Features
 - 사용자에게 머신러닝 피처로 제공되는 암호화된 주식시장 시그널
 - 데이터셋은 소수의 피처를 여러 변형으로 구성
 - 일반적으로 이에 대한 노출을 페널티하지만 100% 중립화는 항상 아니며 예외가 존재
- V3 Features
 - v3 “supermassive” 데이터셋에 사용된 모든 피처
- V4 Medium Safe Features
 - v4 데이터셋에는 5가지 피처 변형이 있다
 - 그중 2가지 변형만 이 서브셋에 포함

타깃 (Targets)

- weekdays
 - 월-금 (20D = 20 Days = 4주)
- returns lag
 - 수익률 계산을 시작하기 전 건너뛴 일수
 - (2L = 2 Lag = 평일 2일 건너뛴)
- timeline XDYL
 - X개의 평일에 걸친 점수, Y일의 지연을 둔 수익률
- neutralizers
 - 타깃이 중립화되는 팩터/피처
- bins=x
 - 타깃의 값을 x개의 구간으로 비닝
- uniformity = x, y, z, ...
 - 바깥쪽 2개 구간(예: 0과 1)에 x%
 - 그다음 안쪽 2개 구간(예: 0.25와 0.75)에 y%
 - 그다음 안쪽 구간(예: 0.5)에 z%
 - ...
- target_[name]_20
 - 타임라인: 20D2L
 - bins=5, uniformity=10%, 40%, 50%
 - 중립화 대상: Common Factors 및/또는 Features

- target_[name]_60
 - 타임라인: 60D2L
 - bins=5, uniformity=10%, 40%, 50%
 - 중립화 대상: Common Factors 및/또는 Features

메타 모델 (Meta Models)

메타 모델은 제출물을 집계해 Numerai가 거래에 사용하는 단일 시그널로 만든다:

- Stake-Weighted Meta Model (SWMM)
 - Numerai 제출물의 스테이크 가중 평균
 - Numerai 헤지펀드가 거래에 사용
- Benchmark Meta Model (BMM)
 - 벤치마크 모델들의 스테이크 가중 평균

점수 (Scores)

- data lag
 - 벤더가 수익률 데이터를 처리하는 데 걸리는 일수
 - 라운드 마감 후 returns lag + data lag 일이 지난 시점부터 점수가 시작(보통 2+2=4일)
- MMC - Meta Model Contribution
 - 제출물, SWMM, target_cyrus_20의 상관 기여도
 - 타임라인: 20D2L (+ 데이터 지연 2일)
- CORR20v2 - Correlation 20D2L v2
 - 제출물과 target_cyrus_20 간의 numerai corr
 - 타임라인: 20D2L (+ 데이터 지연 2일)
- CORJ60 - Correlation Jerome 60D2L
 - 제출물과 target_jerome_60 간의 numerai corr
 - 타임라인: 60D2L (+ 데이터 지연 2일)
- BMC - Benchmark Model Contribution
 - 제출물, BMM, target_cyrus_20의 상관 기여도
 - 타임라인: 20D2L (+ 데이터 지연 2일)
- FNCV3 - Feature Neutral Correlation V3
 - 제출물, V3 Features, target_nomi_20에 대한 feature neutral corr
 - 타임라인: 20D2L (+ 데이터 지연 2일)

- CWMM – Corr w/ Meta Model
 - s^* = tie-kept rank → 가우시안화 → 제출물 s 에 pow 1.5 적용
 - s^* 와 SWMM 간 피어슨 상관 계산
 - 타임라인: 데이터 지연 4일 / 수익률에 비의존
- MCWNM – Max Corr w/ Numerai Models
 - 제출물과 다른 토너먼트 제출물 중 **최대** 피어슨 상관
 - 같은 라운드에서 이루어진 제출들과만 비교
 - 타임라인: 데이터 지연 4일 / 수익률에 비의존
- APCWNM – Average Pairwise Corr w/ Numerai Models
 - 제출물과 다른 각 토너먼트 제출물 간 **평균** 피어슨 상관
 - 같은 라운드에서 이루어진 제출들과만 비교
 - 타임라인: 데이터 지연 4일 / 수익률에 비의존

상관관계 (CORR)

CORR란 무엇인가?

Numerai의 가장 단순한 점수 지표는 모델의 예측과 타깃 간의 상관관계다.

Numerai는 이를 변형한 특별한 상관 지표를 사용하며, 이를 “Numerai Corr”라고 부른다.

높은 수준에서 이 지표는 예측이 실시간 거래에 사용되었을 때의 **실제 포트폴리오 수익률**을 잘 근사하도록 설계되었다.

CORR는 어떻게 계산되는가?

```
def numerai_corr(preds, target):
    # rank (keeping ties) then Gaussianize predictions to standardize
    # prediction distributions
    ranked_preds = (preds.rank(method="average").values - 0.5) /
    preds.count()
    gauss_ranked_preds = stats.norm.ppf(ranked_preds)

    # make targets centered around 0
    centered_target = target - target.mean()

    # raise both preds and target to the power of 1.5 to accentuate the
    # tails
```

```

preds_p15 = np.sign(gauss_ranked_preds) * np.abs(gauss_ranked_preds)
** 1.5
target_p15 = np.sign(centered_target) * np.abs(centered_target) ** 1.5

# finally return the Pearson correlation
return np.corrcoef(preds_p15, target_p15)[0, 1]

```

- 먼저 예측값을 **가우스 순위화(gauss rank)** 한다.
이는 메타 모델이 여러 예측을 앙상블하기 전에 모든 모델의 예측값을 표준화하는 실제 거래 과정과 동일하게 맞추기 위함이다.
- 그다음 타겟 값을 **0을 중심으로(centered)** 정렬한다.
이는 가우스 순위화된 예측값이 평균 0을 기준으로 분포하기 때문에 일관성을 유지하기 위함이다.
- 마지막으로, 가우스 순위화된 예측값과 중심화된 타겟 모두에 **1.5제곱(power of 1.5)** 을 적용한 뒤 피어슨 상관(Pearson correlation)을 계산한다.
이는 헤지펀드가 실제로 가장 높은 혹은 가장 낮은 예측 수익률의 주식만 거래하는 경향이 있기 때문에, **분포의 꼬리(tail)** 를 강조하기 위함이다.

핵심 요점은 예측값의 **분포 자체는 중요하지 않다**는 것이다.

모델은 오직 예측의 **순위(rank)** 에 의해서만 평가된다.

또한, 일반적인 순위 상관(rank-correlation)보다 **꼬리 부분의 정확도**가 점수에 더 큰 영향을 미친다.

웹사이트상의 CORR

웹사이트에서는 여러 형태의 CORR 점수를 확인할 수 있다:

- CORR20V2 – 메인 타겟의 20일 버전에 대한 최신 CORR 점수
- CORJ60 – 보조 타겟 “Jerome”의 60일 버전에 대한 상관관계
- CORT20 – 보조 타겟 “Teager”의 20일 버전에 대한 상관관계

메타 모델 기여도 (Meta Model Contribution, MMC)

MMC (및 BMC)란 무엇인가?

Meta Model Contribution (MMC) 은 모델의 예측값을 메타 모델에 대해 중립화한 후, 그 결과가 타깃과 가지는 **공분산(covariance)** 을 의미한다.

유사하게, **Benchmark Model Contribution (BMC)** 은 모델의 예측을 **스테이크 가중 벤치마크 모델(Benchmark Models)** 에 대해 중립화한 뒤 타깃과의 공분산을 측정한다.

이 두 지표는 모델의 **고유한 정보(unique component)** 가 메타 모델(또는 BMC의 경우 벤치마크 모델)의 상관관계에 얼마나 기여하는지를 나타낸다.

즉, 모델의 예측을 메타 모델(또는 벤치마크 모델)에 대해 중립화(neutralize)하면, 남은 **직교(orthogonal)** 성분이 타깃과 가지는 공분산이 해당 모델의 기여도다.

계산 방식 (Calculation)

특정 라운드에서 사용자의 MMC를 계산하는 절차는 다음과 같다:

1. 제출된 예측값(predictions)을 정규화(normalize)한다.
2. 메타 모델을 정규화한다.
3. 사용자의 제출 예측을 메타 모델에 대해 중립화한다.
4. 중립화된 예측과 타깃 간의 공분산(covariance)을 계산한다.

```
def contribution(
    predictions: pd.DataFrame,
    meta_model: pd.Series,
    live_targets: pd.Series,
) -> pd.Series:
    """Calculate the contributive correlation of the given predictions
    wrt the given meta model.

    Then calculate contributive correlation by:
    1. tie-kept ranking each prediction and the meta model
    2. gaussianizing each prediction and the meta model
    3. orthogonalizing each prediction wrt the meta model
    4. multiplying the orthogonalized predictions and the targets

    Arguments:
        predictions: pd.DataFrame - the predictions to evaluate
        meta_model: pd.Series - the meta model to evaluate against
        live_targets: pd.Series - the live targets to evaluate against

    Returns:
        pd.Series - the resulting contributive correlation
                     scores for each column in predictions
    """
```

```

# filter and sort preds, mm, and targets wrt each other
meta_model, predictions = filter_sort_index(meta_model, predictions)
live_targets, predictions = filter_sort_index(live_targets,
predictions)
live_targets, meta_model = filter_sort_index(live_targets,
meta_model)

# rank and normalize meta model and predictions so mean=0 and std=1
p = gaussian(tie_kept_rank(predictions)).values
m = gaussian(tie_kept_rank(meta_model.to_frame()))
[meta_model.name].values

# orthogonalize predictions wrt meta model
neutral_preds = orthogonalize(p, m)

# center the target
live_targets -= live_targets.mean()

# multiply target and neutralized predictions
# this is equivalent to covariance b/c mean = 0
mmc = (live_targets @ neutral_preds) / len(live_targets)

return pd.Series(mmc, index=predictions.columns)

```

진단에서의 BMC (BMC in Diagnostics)

진단(디아그노스틱스)에서는 BMC가 스테이크 가중 벤치마크 모델 전체에 대해 계산되지 않는다. 대신, 여기에 명시된 것처럼 **가장 많은 스테이크를 보유한 단일 벤치마크 모델**에 대해 계산된다.

리더보드(Leaderboard) 상의 BMC와 진단 내 BMC가 다른 이유는 다음과 같다:

- 리더보드(Leaderboard)** 는 **실시간 성과(live performance)** 를 보여준다.
 즉, 리더보드에서 계산된 BMC는 당시 Numerai와 데이터 사이언티스트들이 **그 시점에 알고 있던 정보**를 기반으로 한다.
 따라서 특정 시점의 스테이크 가중 벤치마크 모델에 대해 평가하는 것이 공정하다.
 예를 들어, 초기 라운드에서는 벤치마크 모델이 v2 예측(example predictions)뿐이었지만, 최근 라운드에서는 훨씬 정교한 모델들이 포함되어 있다.
- 진단(Diagnostics)** 은 **검증 성과(validation performance)** 를 보여준다.
 따라서 더 나은 데이터, 타깃, 모델링 기법을 사용하고 있을 수 있다.
 그런 상황에서 오래된 v2 예측을 기준으로 평가하는 것은 적절하지 않다.
 대신, 가능한 한 최신의 고성능 모델을 기준으로 평가하는 것이 타당하다.

리더보드와 진단에서 BMC가 왜 다른지 여전히 명확하지 않다면,
이 개념을 다루는 [타겟 앙상블 노트북\(target ensemble notebook\)](#)을 참고하라.

토론 (Discussion)

MMC 및 BMC에 대한 더 자세한 논의는 [이 포럼 글](#)에서 확인할 수 있다.

그랜드마스터 & 시즌 (Grandmasters & Seasons)

{% embed url="https://www.youtube.com/watch?v=R81Bdu2Suoo" %}

그랜드마스터(Grandmasters)는 Numerai 참가자들을 위한 명예 등급 제도다.
그랜드마스터 시스템에서는 사용자의 **스테이크 가중 모델 성과(stake-weighted model performance)**를 기준으로 연 단위(캘린더 기준) 평가가 이루어진다.

매년 1월 1일, 새로운 시즌이 시작된다.
데이터 사이언티스트는 **20회의 유효 제출(qualifying submissions)**을 완료하면
해당 연도의 **누적 스테이크 가중 계정 점수 순위**를 기반으로 임시 타이틀(provisional title)이 부여된다.
이 타이틀은 연중 성과에 따라 변동될 수 있다.

시즌의 마지막 라운드가 종료되고 순위가 확정되면, 최종 타이틀이 공식적으로 수여된다.

캐논 점수 (Canon Scores)

‘캐논 점수(Canon Scores)’란 서로 다른 버전의 점수 체계를 하나의 일관된 지표로 통합하기 위해 사용되는 **표준화된 메트릭**을 의미한다.

Numerai 토너먼트의 맥락에서 Canonical Scores(또는 “Canon Scores”)는 매우 중요한 개념이다.
예를 들어, 보상 지표로 사용되는 CORR 점수는 여러 차례 업데이트되었다.
초기에는 **‘CORR20’**이 보상 점수로 사용되었으나,
라운드 **484**까지 적용된 후 **485 라운드부터 ‘CORR20v2’**로 진화했다.
‘Canon CORR’ 점수는 이러한 변화를 통합하여,
484 라운드까지는 ‘CORR20’을, 485 라운드 이후에는 ‘CORR20v2’를 사용한 **통합 점수**로 정의된다.

‘TC’ 점수는 토너먼트 역사 전반에 걸쳐 동일하게 유지되었기 때문에,
‘Canon TC’와 ‘TC’는 동일한 개념이다.

2022년 및 2023년 시즌에서는 **Canon TC**와 **Canon CORR** 점수가 기준으로 사용되었다.
하지만 향후 시즌에서는 점수 산정 기준이 변경될 수 있다.

스테이크 가중 평균 점수 (Stake-weighted Average, SWA Scores)

스테이크 가중 점수 이해하기 (Understanding Stake-weighted Scores)

그랜드마스터 랭킹에서 참가자의 성과는 단순히 모델의 원점수(raw score)만으로 평가되지 않는다. 참가자가 자신의 모델에 얼마나 **확신(confidence)** 을 가지고 있는지를 나타내는 **스테이크(stake)** 또한 중요한 요소다. 이 두 요소를 결합한 지표가 바로 **스테이크 가중 평균 점수(stake-weighted average score)** 다. 이는 모델의 정확도와 스테이크 규모를 함께 고려하여 보다 종합적인 성과를 측정한다.

2022년 및 2023년 시즌에서 참가자들의 순위는 **Canon CORR** 및 **Canon TC** 메트릭을 기반으로 한 **스테이크 가중 평균 점수(SWA)** 에 의해 결정되었다.

시즌 점수 계산 (Season Scoring)

시즌 점수는 해당 시즌에 포함된 모든 라운드의 **스테이크 가중 평균 점수의 평균값**으로 계산된다.

어떤 라운드에서 모델을 제출하지 않거나 스테이킹을 하지 않은 경우, 그 라운드의 점수는 **0점으로 처리**되며, 이는 전체 시즌 평균에 부정적인 영향을 미친다.

예시 (Example)

예를 들어, 한 라운드에서 두 개의 스테이킹된 모델과 하나의 비스테이킹 모델을 제출했다고 하자:

- 모델 A – 점수 0.01, 스테이크 10 NMR
- 모델 B – 점수 0.02, 스테이크 5 NMR
- 모델 C – 점수 0.03, 스테이크 없음

이 경우, 해당 라운드의 **스테이크 가중 평균 점수**는 다음과 같이 계산된다:

$$\text{SWA score} = ((0.01 * 10) + (0.02 * 5)) / (10 + 5) = 0.0167$$

모델 C는 스테이킹되지 않았기 때문에 계산에 포함되지 않는다.

순위 및 등급 (Rankings and Tiers)

Numerai 토너먼트 참가자들은 성과에 따라 다음과 같은 **등급(title)** 을 부여받는다.
시즌의 마지막 라운드가 종료되고 순위가 확정되면, 아래 기준에 따라 타이틀이 수여된다:

- **Grandmasters** – 1위
- **Masters** – 상위 10위
- **Experts** – 상위 100위
- **Researchers** – 상위 1/3
- **Contributors** – 중간 1/3
- **Apprentices** – 하위 1/3
- **Novices** – 20회의 유효 제출을 아직 완료하지 않은 참가자

참가자는 여러 점수 카테고리에서 독립적으로 평가되므로, 여러 타이틀을 동시에 얻을 수도 있다.
예를 들어, 한 참가자가 어떤 점수 부문에서는 "Expert" 등급을, 다른 부문에서는 "Researcher" 등급을 가질 수 있다.

자격 요건 (Qualification)

그랜드마스터 시즌 자격은 **활발하고 지속적인 참여**를 기반으로 한다.
시즌에 포함되려면 **20회의 유효 라운드(Qualified Rounds)** 를 완료해야 한다.
라운드는 다음 조건을 충족할 때 **유효 라운드(Qualified Round)** 로 간주된다:

1. 해당 라운드에 대해 **정시 제출(on-time submission)** 이 있을 것
2. 제출된 모델들에 걸린 **총 스테이크 NMR이 1 이상일 것**

예를 들어, 단일 모델에 1 NMR을 스테이킹하고 20회(약 한 달간) 정시 제출을 완료하면,
시즌 자격을 공식적으로 얻게 된다.
이 조건을 충족하면 시즌 랭킹에 포함되어 성과에 따른 타이틀을 받을 수 있다.

시즌 자격 충족 요건 (Earning Your Place in the Season)

- **최소 스테이크 기준(Minimum Stake Threshold):**
모든 모델의 합산된 위험 노출 스테이크(at-risk stake)가 라운드당 **1 NMR 이상**이어야 한다.
이는 모든 참가자가 토너먼트 결과에 실질적인 이해관계를 갖도록 보장한다.
- **유효 라운드(Qualified Round):**
라운드가 유효로 인정되기 위해서는 최소 스테이크 기준을 충족하는 스테이킹된 모델로
정시 제출(on-time submission) 을 해야 한다.

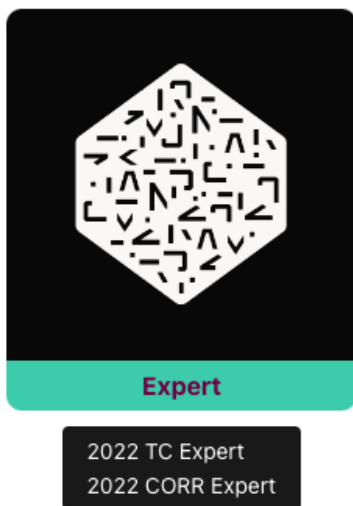
- **고유 라운드(Distinct Rounds):**

시즌 자격을 얻기 위해서는 시즌 기간 동안 서로 다른 20개의 라운드에서 유효 제출을 해야 한다. 동일 라운드 내 여러 제출은 하나로 간주되며, 중복 제출은 인정되지 않는다.

자격 충족 후 (What's Next After Qualification)

이 요건을 충족하면 시즌 랭킹에 포함되어, 성과에 따라 타이틀이 부여된다.

획득한 타이틀 (Earned Titles)



NUMERAI BENCHMARK MODELS

San Francisco, CA · Joined 5 years ago

Submits example scripts and benchmark models developed by Numerai.

https://docs.numerai.ai/numerai-tournament/benchmark_models

타이틀을 획득하면 다음과 같은 방식으로 표시되어 업적을 기념한다:

- **프로필 하이라이트(Profile Highlight):**

리더보드에서 프로필 사진 주위에 색상 테두리로 가장 높은 타이틀이 표시된다. 각 타이틀은 고유 색상으로 구분되어 한눈에 식별 가능하다.

- **계정 프로필(Account Profile):**

계정 프로필 페이지에서도 타이틀이 표시된다.

- **세부 정보 톨팁(Detailed Insights):**

타이틀 위에 마우스를 올리면, 지금까지 획득한 모든 타이틀 정보를 보여주는 톨팁이 표시된다.

- **디스코드 연동(Discord Integration):**

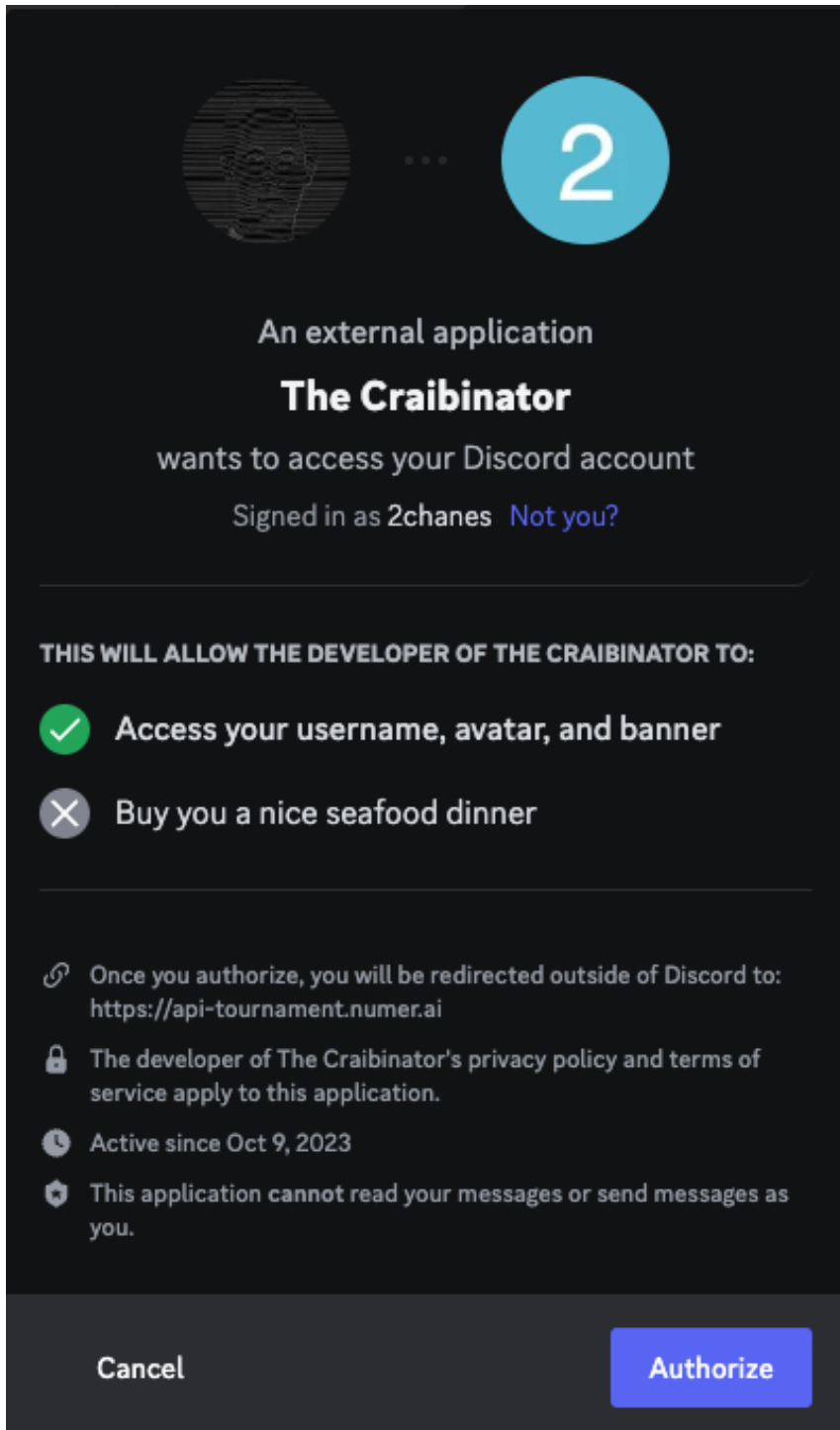
Numerai 계정을 Discord와 연동하면, 가장 높은 타이틀이 Discord 프로필의 역할(role)로 표시된다.

Discord 연동 (Discord Integration)

이제 사용자는 자신의 Numerai 계정을 Discord 계정과 연동해, 그랜드마스터 타이틀에 해당하는 Discord 역할(role)을 받을 수 있다.

Discord 계정 연결 방법

1. 계정 프로필로 이동하여 **“Edit account profile”** 버튼을 클릭한다.
2. **“Connect Discord account”** 버튼을 클릭한다.
3. Discord 인증 페이지로 리디렉션된다.
4. Numerai Discord 봇 **‘The Craibinator’** 의 연결을 승인(Authorize)한다.



연동이 완료되면, Numerai에서 획득한 타이틀이 자동으로 Discord 계정의 역할(role)로 반영된다. 또한 Discord 프로필이 Numerai 프로필 페이지에 표시되어, 커뮤니티 내에서의 존재감이 강화된다.

스테이킹 (Staking)

소개 (Introduction)

모델의 성과에 자신이 생겼다면, Numerai의 암호화폐 NMR로 해당 모델에 스테이킹할 수 있다.

모델이나 제출물에 대한 스테이킹이란, 제출의 채점 기간 동안 **NMR을 잠금(lock-up)** 상태로 두는 것을 의미한다.

Numerai에 스테이킹하면, 스테이킹된 기간 동안 Numerai는 사용자의 스테이크에 대해 보상을 지급하거나 소각할 권한을 갖는다.

스테이킹된 제출의 채점이 완료되면,
양의 점수를 얻은 경우 NMR 보상을 받으며,
음의 점수를 얻은 경우 스테이킹된 일부 NMR이 **소각(burned)** 된다.

이 시스템은 두 가지 중요한 기능을 수행한다:

1. **“Skin in the game”** – Numerai는 스테이킹된 예측의 품질을 신뢰할 수 있다.
2. **보상과 소각(payouts and burns)** – SWMM의 가중치를 지속적으로 개선한다.

 스테이킹 기회는 투자 계약, 증권, 파생상품, Numerai의 헤지펀드 또는 수익참여를 의미하지 않는다.

Numerai 또는 그 수수료 구조에 대한 투자 기회를 기대하는 사용자는 스테이킹하지 말아야 한다.

자세한 내용은 이용약관(Terms of Service)을 반드시 참고하라.

Numeraire (NMR)

NMR은 Numerai의 스테이킹 및 보상 시스템을 구동하는 암호화폐다.

- NMR에 대한 자세한 내용은 Github에서 확인할 수 있다: <https://github.com/erasureprotocol/NMR>
- 토큰 통계는 여기서 볼 수 있다: <https://numer.ai/nmr>

모델에 NMR을 스테이킹하려면, 먼저 NMR을 확보한 후 이를 Numerai 지갑에 입금해야 한다.

필요 시 해당 지갑에서 ETH 주소로 출금할 수도 있다.

NMR 구매 및 입금 (Depositing NMR)

다음 두 가지 방법을 추천한다:

- [Coinbase](#) – USD, GBP, EUR 또는 비트코인으로 NMR을 손쉽게 구매할 수 있는 안전한 거래소. 암호화폐 초보자에게 적합하며, 지역에 따라 이용 가능 여부가 다르다.
- [Uniswap](#) – ETH나 USDC 등 이더리움 기반 토큰을 NMR로 교환할 수 있는 탈중앙화 거래소. MetaMask 같은 개인 지갑이 필요하며, DeFi 사용자에게 적합하다.

NMR을 구매한 후, [Wallet](#) 주소를 사용해 Numerai 지갑으로 전송한다:

Available balance

0.440808483525612275 NMR

Deposit

Withdraw

Send NMR to

0x000000...000215ea 

Click to copy your deposit address

The deposit address only supports NMR. Anything else sent to it will be lost forever.

Get NMR

제한 사항 (Restrictions)

사기 및 자금세탁 방지를 위해, 신규 계정에는 출금 제한이 있다. Numerai 지갑에서 출금하려면 다음 중 하나를 만족해야 한다:

1. 계정 생성 후 **30일 이상 경과했을 것**,
2. 또는 **0.1 NMR 초과 금액**을 출금할 것.

즉, 계정을 막 생성하고 0.1 NMR을 입금한 경우, 계정이 30일 이상이 되거나 더 많은 NMR을 추가 입금하기 전까지는 출금할 수 없다.

스테이크 증가 (Increasing Stake)

numer.ai/staking 페이지로 이동한 후, 스테이킹을 원하는 모델 옆의 “**manage stake**” 버튼을 클릭한다:

Models (56)											Search	
NAME	STAKE	CORR20v2	TC	MMC	MULTIPLIERS	PAYOUT	1D	3M	1Y	ALL		
 INTEGRATION_TEST	8.41 NMR	0.0069	0.0000	0.0001	0.5XCORR20v2 2XMMC	Compound	▲ 0.97%	▲ 21.9%	▲ 38.4%	▲ 877.7%	확인	

화면이 작은 경우, 이 버튼을 찾기 위해 스크롤해야 할 수도 있다.

그러면 **스테이크 모달(Stake Modal)** 이 열리며,
지갑에 보유한 NMR을 선택한 모델에 스테이킹할 수 있다:

Stake

Release

Advanced

Stake Amount ⓘ

0.1

NMR

Available: 0.450808483525612275 NMR (max) ⓘ

Confirm

스테이크 해제 (Releasing Stake)

스테이킹된 NMR은 사용자가 직접 해제(release) 요청을 하기 전까지 **잠금(lock)** 상태로 유지된다.
해제 요청 후에는 **1개월**이 지나야 지갑으로 반환된다.
정확히는, 해제 요청 시점의 **마지막 활성 라운드**가 **종료(resolution)** 될 때 NMR이 반환된다.

해제 대기 중인 금액은 향후 지급(payout)에 포함되지 않지만,
여전히 **소각(burn)** 대상이 될 수 있다.

Stake

Release

Advanced

Release Amount ⓘ

Release Date: 2023-05-03 ⓘ

Amount

NMR

Available: 0.2 NMR (max)

☐ Drain

Confirm

보상 (Payouts)

사용자의 **보상(payout)** 은 기본적으로 **점수(score)** 에 따라 결정된다.

양의 점수를 얻으면 보상을 받고,

음의 점수를 얻으면 스테이킹된 NMR의 일부가 **소각(burn)** 된다.

라운드별 최대 보상 또는 소각 비율은 **±5%로 제한**되며,

다음 공식을 기반으로 계산된다:

```
payout = stake * clip(payout_factor * (corr * 0.5 + mmc * 2), -0.05, 0.05)
```

stake 는 해당 라운드의 **마감(close)** 시점에서 모델에 스테이킹된 NMR의 총액을 의미한다.

이는 라운드별로 위험에 노출된 스테이크 값(stake value at-risk)이라고도 부른다.

특정 라운드의 stake value at-risk 에는

해제 대기 중인 미스테이킹(unstaked) 금액이 포함되지 않는다.

또한, 라운드에 유효한 제출이 없을 경우 이 값은 **0으로 설정**된다.

보상 계수 (Payout Factor)

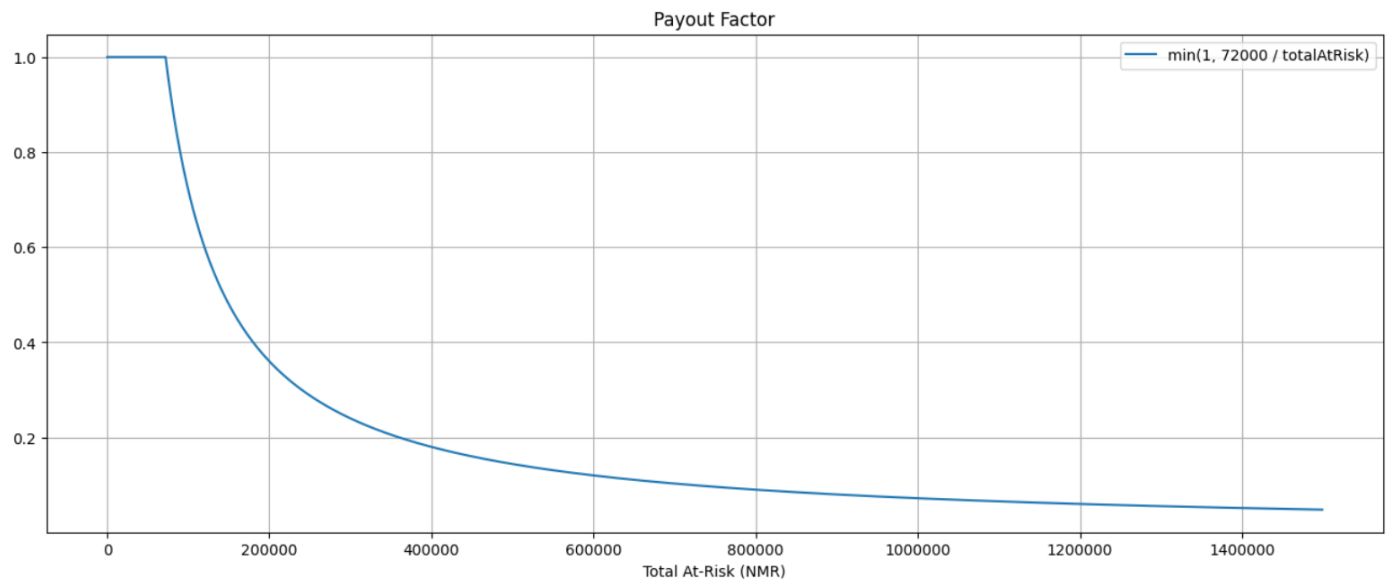
payout_factor 는 총 스테이킹된 NMR 규모에 따라

스테이킹 한도(staking_threshold)와 반비례적으로 조정되는 동적 값(dynamic value) 이다.

```
payout_factor = min(1, staking_threshold / total_at_risk)
```

Tournament	Stake Threshold
Numerai	72000
Signals	36000
Crypto	10000

다음은 스테이크가 증가함에 따라 Numerai의 payout factor가 어떻게 보이는지에 대한 예시다:



NMR 출금 (Withdrawing NMR)

Numerai 지갑에서 NMR을 출금할 준비가 되면 [Wallet Page](#)로 이동하여 금액을 지정하고, 목적지 주소를 해당 입력란에 붙여넣는다:

29.000000000000000000000000 NMR

Withdraw

NMR

[illegible]

Next

마지막으로 “Next”를 클릭하고, 확인 메시지가 표시될 때까지 기다린다. 트랜잭션이 완료되는 데 몇 분 정도 소요될 수 있으니, 잠시만 기다려 달라.

세금 보고 (Tax Reporting)

우리는 보통 전년도에 대한 세금 보고서를 1월 중순에 발행한다. 오른쪽 상단의 계정 아이콘에 마우스를 올리
고 > “Settings”를 클릭 > “TAXES & REPORTS” 섹션에서 확인할 수 있다.

바운티 (Bounties)

우리는 Numerai를 개선하기 위해 여러분의 도움이 필요하며, 바운티를 명확하고 공정하게 운영하고자 한다. 아래의 예시는 포괄적이지 않으며, 바운티 금액은 대략적인 가이드라인일 뿐이다 — 실제 금액은 버그, 피드백 또는 익스플로잇의 영향과 난이도에 따라 달라진다. **바운티 지급 여부 및 금액은 전적으로 Numerai의 재량에 따라 결정된다.**

보고에 대한 응답에는 영업일 기준 3–5일이, 보고된 문제를 해결하는 데에는 추가로 7–10영업일이 걸릴 수 있다. 우리는 작은 팀이며 영향이 큰 프로젝트를 우선시해야 하므로, 부디 인내심을 부탁드립니다 — 하나의 보고에 대해 1회 이상 재촉 메시지를 보내지 말아 달라.

바운티를 받으려면 [Numerai Tournament](#) 계정이 필요하다. 미국 거주자가 \$600 USD를 초과하는 바운티를 받는 경우 [W9 납세자 정보](#)를 제출해야 한다. **세법 관할과 무관하게, 바운티 지급과 관련된 모든 세무상 책임은 전적으로 수령자 본인에게 있다.**

버그 (Bugs)

무언가가 고장 난 것을 발견하면 신고해 달라! 실제 문제로 판명되고, 귀하의 보고가 문제 해결에 도움이 되었다면 바운티를 지급한다:

버그 리포트	바운티
웹사이트의 경미한 표시/스타일 이슈, 잘못된 이메일, 끊긴 링크, 오타자	0.1–1 NMR
잘못된 데이터/채점/보상 또는 서비스 장애(예: 제출/스테이킹 불가)	1–5 NMR

보안 익스플로잇 & 취약점 (Security Exploits & Vulnerabilities)

Numerai는 보안을 매우 중요하게 여기지만, 작은 팀이므로 익스플로잇 및 취약점의 우선순위, 심각도, 영향을 특히 중시한다:

취약점 보고	바운티
Numerai 또는 사용자 자금이 위험을 초래하지 않는 저영향 익스플로잇/취약점	0.1–2 NMR
상황에 따라 Numerai 또는 사용자 자금이 위험을 초래할 수 있는 중간 영향의 익스플로잇/취약점	2–10 NMR
비교적 쉽게 Numerai 또는 사용자 자금이 위험을 초래할 수 있는 고영향 익스플로잇/취약점	10–100 NMR
Numerai 및 스테이커의 모든 자금이 위험을 초래하여 토너먼트를 사실상 종료시킬 수 있는 중대한 보안 위험	100+ NMR

연구를 시작하기 전에, 다음과 같은 행위는 자제해 주길 바란다:

- 스팸, 서비스 거부(DoS), 또는 Numerai 운영 서비스에 영향을 주는 행위
- 과도한 요청(레이트 리미트 공격) — 단, 매우 낮은 임계치로 심각한 위험이 입증되는 경우는 예외
- 다른 사용자의 계정을 공격하거나 탈취하는 행위
- 익스플로잇이나 취약점을 공개적으로 공유하는 행위

고우선 (High Priority)

다음 이슈들은 심각한 보안 우려로 간주한다:

- 사용자의 직접적인 조작 없이 **사용자 계정에 접근** 가능
- **API 토큰, 웹사이트 자격증명 또는 기타 민감 정보의 유출**
 - 자격증명 유출 보고 시, 실제로 기존 Numerai 계정에 로그인 가능한 **유효한 이메일/비밀번호** 조합을 포함해야 한다.
 - 이 자격증명은 ["email", "password"] 컬럼을 갖춘 **유효한 CSV 파일**로 제출해야 한다. 다른 방식으로 제출된 자격증명은 접수/검증할 수 없다.
 - 보상 금액은 다음 기준에 따른다:
 - 계정에서 NMR을 실제로 탈취할 수 없다면 “저영향”으로 간주하며, 누출된 계정당 **0.1 NMR**을 지급한다.
 - **실제로** 계정에서 NMR 탈취가 가능하다면, 계정의 **전체 스테이크**에 상응하는 보상을 지급할 수 있다.
- Numerai 또는 사용자 자금이 **상호작용 없이** 손실되는 경우
- Numerai의 운영 서비스에 Numerai의 상호작용 없이 접근 또는 제어 권한을 획득
- 의도치 않은 보상으로 이어질 수 있는 토너먼트 익스플로잇

- 서브도메인 탈취 — 페이지를 점유할 수 있음을 **비공격적 메시지(예: 사용자명)** 로 증명해 달라

일반적으로 위 항목들은 3-5영업일 내에 해결하며, 바운티는 상단 범위로 책정되는 경향이 있다.

저우선 (Low Priority)

다음 항목들은 보안 영향이 미미한 이슈로 간주한다. 명확하고 중대한 익스플로잇이 쉽게 재현되지 않는 한 보고를 자제해 달라. 보고 시에는 Numerai 및 사용자에게 어떤 피해를 줄 수 있는지에 대한 **구체적 설명**과 **유효한 PoC**가 필요하다:

- 자동 스캐너/툴(Acunetix, Vega, OSINT Framework 등)이 보고하나, 명확한 익스플로잇이 없는 취약점
- 구성/인증서 누락이 있으나 Numerai 또는 사용자를 쉽게 해칠 수 없는 경우:
 - 보안 헤더 누락
 - SSL/TLS 스캔 보고서(예: SSL Labs)
 - DNS 구성/인증서 누락
 - 오픈 포트
 - 연계되지 않은 오픈 리다이렉트
 - 프로토콜 불일치
 - 레이트 리미팅
 - 노출된 로그인 패널
 - Dangling IP
 - 인증이 필요 없는 쿠키의 보안 플래그 누락
 - 보안 영향이 경미한 CSRF(예: 로그아웃 CSRF)
- 내부 구조 노출에 관한 우려이나, 실제 공격에 대한 구체적 증거 없이 **이론적 공격**만 제기하는 경우:
 - 스택 트레이스
 - 경로 노출
 - 디렉터리 리스팅

이들 이슈는 심각도에 따라 **5-20영업일**이 걸릴 수 있으며, 바운티는 하한에 가까울 가능성이 높다.

무우선/범위 외 (No Priority / Out-of-Scope)

다음 사례들은 취약점으로 간주하지 않으며, 바운티 대상이 아니다:

- 단계별 PoC 없이 **추측성**이 강하거나 **이론적 피해**만 제기하는 보고
- Numerai 또는 타 사용자에게 **실질적 악용**으로 이어지기 어려운 취약점(예: self-XSS, 콘솔에 JS 붙여넣기)

- 모범사례/업계 표준 관련 우려 — 장기적으로 중요하나, 주요 공격 벡터가 아니며, 우리는 계정/자금/민감 데이터의 침해 경로에 집중한다
- 구형 브라우저/앱 버전에만 영향을 미치는 취약점 — 최신 Safari/Firefox/Chrome/Edge/IE 버전에 대한 익스플로잇만 고려
- 다음과 같은 오류/오류 메시지/스택 트레이스:
 - 500 내부 서버 오류에서 기인
 - 민감 정보를 노출하지 않음
- 비밀번호 정책 및/또는 무차별 대입 공격
- 레이트 리밋을 우회하지 않는 모든 형태의 무차별 대입 공격
- 레이트 리밋을 우회하지 않는 사용자 열거(user enumeration)
- “피해자” 이메일로 가입하여 해당 이메일의 계정 생성을 차단하는 행위
- 이메일 별칭(alias)으로 가입
- 임시/일회용 이메일로 가입
- API를 통한 **비실행 파일**(CSV, 이미지, parquet 등) 업로드
- API에 나열된 **공개 파일**(CSV, 이미지, parquet 등) 다운로드
- 과도/불필요한 데이터 저장(예: 오래된 업로드)
- 신뢰된 제3자(예: Google Analytics 등)에게 토큰 전송
- 무작위로 생성되어 추적이 불가능한 공개 URL
- numer.ai, signals.numer.ai, crypto.numer.ai, forum.numer.ai 등 **공개 웹사이트**에서 사용하는 **비민감 공개 API**의 정보
 - GraphQL 인트로스펙션 포함
- 이미 인증이 필요한 행위에 대해 비밀번호 재입력이 없는 경우
- 사용자의 부주의로 인한 문제:
 - 사용자 제공 링크(프로필 바이오 링크, 포럼 링크 등) 클릭
 - 피해자 기기/컴퓨터에 물리적으로 접근
 - 피싱, 사회공학, 공용 컴퓨터 관련 우려
- 우리 포럼 관련:
 - 포럼에서 기본 HTML 태그(예: <h1>, <a> 등) 사용
 - Numerai 포럼 자체의 보안 이슈 — Discourse 플랫폼을 사용하므로 해당 이슈는 Discourse에 보고해야 한다. 자세한 내용은 [여기](#)를 참고

일반적으로, 위 이슈들의 해결에는 **5-20영업일**이 걸릴 수 있으며, 바운티는 하한에 가까울 가능성이 높다.

익스플로잇 & 취약점 보고 방법 (How to report Exploits & Vulnerabilities)

보고서에는 다음이 포함되어야 한다:

- 익스플로잇 또는 취약점에 대한 설명
- Numerai 또는 사용자에게 초래할 수 있는 **구체적 영향**
- 익스플로잇/취약점이 얼마나 **쉽게 악용될 수 있는지**에 대한 논의
- **단계별 가이드**가 포함된 공격 실행 방법
- **PoC 영상** — 프로덕션 서비스나 타 사용자에게 영향을 주는 공격인 경우, 영상이 없으면 바운티 대상에서 제외

보고서를 모두 준비했다면, 제목을 “[Security Report] 보고서의 간결한 제목”으로 하여 security@numer.ai 로 이메일을 보내 달라.

피드백 및 제안 전달 (How to give feedback and suggestions)

[Discord](#)에서 메시지를 보내 달라.

큰 바운티를 목표로 하는 경우, 아이디어를 문서(pdf 또는 구글 문서)나 노트북(Colab, GitHub) 형태로 정리해 주면 도움이 된다.

Numerai 시그널즈 (Numerai Signals)

개요 (Overview)

소개 (Introduction)

[Numerai Tournament](#)가 무료 데이터를 제공하는 반면, [Numerai Signals](#)은 사용자가 **자신만의 데이터**, 즉 **고유한 시그널(signal)** 을 제출하도록 요구한다.

주식시장에서의 **시그널**은 정보의 흐름(feed of information)을 의미한다.

이는 주식에 대한 수치적 데이터로, Numerai 같은 **퀀트 헤지펀드(quantitative hedge fund)** 가 이를 사용해 시장을 모델링하고 포트폴리오(주식 보유 컬렉션)를 구성한다.

다음은 주식시장 시그널의 예시들이다:

- 기초적 신호(Fundamental signals)
예: P/E 비율, 배당수익률, 애널리스트 등급

- 기술적 신호(Technical signals)
예: MACD, RSI, MFI
- 대체 데이터 신호(Alternative data signals)
예: 신용카드 거래 데이터, 위성 이미지, 소셜 미디어 감성 분석
- 블렌디드 신호(Blended signals)
예: Barra 리스크 요인, Fama-French 요인

데이터 제공자는 고유한 **피처(features)** 를 시그널로 직접 제출할 수 있으며, 데이터 사이언티스트는 고유 데이터를 모델링하여 **예측값(predictions)** 을 시그널로 제출할 수 있다. 제출된 시그널들은 Numerai의 타겟(targets) 및 다른 참가자들의 시그널과 비교되어 평가된다. 또한 NMR 암호화폐를 이용해 스테이킹할 수 있으며, 성과에 따라 NMR을 **획득(earn)** 하거나 **소각(burn)** 하게 된다.

Numerai Signals는 Numerai가 구상하는 “세계 마지막 헤지펀드(the world’s last hedge fund)”를 구축하기 위한 마스터 플랜의 일부이다.

자세한 내용은 [Medium 게시물](#) 과 [단편 영상](#)을 참고하라.

데이터 (Data)

다음은 **Signals V1** 데이터를 시작하는 예시다:

```
from numerapi import NumerAPI
import pandas as pd

napi = NumerAPI()
# 저장 공간과 메모리를 절약하기 위해 int8 사용
napi.download_dataset("signals/v2.0/train.parquet")
training_data = pd.read_parquet("signals/v2.0/train.parquet")
```

이 데이터는 주로 사용자의 자체 데이터에 **추가하거나**, 혹은 시그널을 **중립화(neutralize)**할 때 사용하도록 설계되어 있다. 이는 Numerai가 이미 보유하고 있으며, 내부적으로 잘 활용할 수 있는 데이터이기 때문이다.

고품질 시그널을 생성하려면, 독자적이고 고유한 주식시장 데이터를 확보해야 한다.

시작을 위해 사용할 수 있는 다른 데이터 제공업체로는 [Yahoo Finance](#), [Quandl](#), [Koyfin](#) 등이 있다. 또한 [QuantConnect](#)나 [Alpaca](#) 같은 플랫폼을 통해 쉽게 시그널을 생성할 수도 있다.

무료 혹은 저비용 데이터, 플랫폼, 도구 목록은 [이 Numerai 포럼 게시물](#)에서 확인할 수 있다.

만약 자체 주식시장 데이터를 보유하고 있지 않다면, 대신 [Numerai Tournament](#)를 사용하는 것이 좋다.

모델링 (Modeling)

데이터 모델링을 어디서 시작해야 할지 모르겠다면, [이 페이지](#)에 여러 튜토리얼 모델이 준비되어 있다.

또한 Numerai Tournament를 통해 [기초 모델링\(Basic Modeling\)](#)을 학습하는 것을 강력히 추천한다.

모델링 기술에 자신이 생기면, 그 경험은 자연스럽게 Signals에서도 활용될 수 있다.

다음은 Signals 모델의 예시다:

```
import lightgbm as lgb

features = [
    f for f in training_data.columns
    # there are two non-numerical feature cols
    if "feature" in f and f not in ("feature_country",
    "feature_exchange_code")
]

model = lgb.LGBMRegressor(
    n_estimators=2000,
    learning_rate=0.01,
    max_depth=5,
    num_leaves=2 ** 5,
    colsample_bytree=0.1
)
model.fit(
    training_data[features],
    training_data["target"]
)
```

참고:

이 모델은 매우 기초적인 예시이며, Numerai가 이미 보유한 기본 데이터를 기반으로 학습되었기 때문에 실제 성과는 거의 없을 가능성이 높다.

제출 (Submissions)

Signals의 제출 방식은 [Numerai Tournament](#)와 매우 유사하다.
다만, **id** 대신 **주식 티커(stock tickers)** 를 사용한다는 점이 다르다.

사용자의 시그널은 각 주식 티커에 대해 하나의 수치 값을 포함한 리스트 형식이어야 한다.

제출 파일에 포함된 주식 티커 목록은 **Numerai Signals 주식시장 유니버스(Numerai Signals stock market universe)** 에 의해 정의된다.
각 티커의 수치 값은 **0과 1** 사이의 범위에 있어야 한다.

다음은 Python을 사용하여 실시간 예측을 생성하고 업로드하는 예시다:

```
# Use API keys to authenticate
sapi = SignalsAPI("[your api public id]", "[your api secret key]")

# Download latest live features
sapi.download_dataset(f"signals/v2.0/live.parquet")
live_data = pd.read_parquet(f"signals/v2.0/live.parquet")

# Generate live predictions
live_predictions = model.predict(live_data[features])

# Format and save submission
submission = pd.Series(
    live_predictions, index=live_data.index
).to_frame("prediction")
submission.to_csv(f"submission.csv")

# Upload submission
sapi.upload_predictions(f"submission.csv", model_id="your-model-id")
```

Numerai는 **Signals 제출물의 구조**에 대해 여러 가지 요구사항을 가지고 있다.
모든 요건을 완전히 이해하려면 [이곳](#)을 참고하라.

채점 (Scoring)

Signals의 채점 방식은 [Numerai Tournament](#)와 매우 유사하다.
가장 큰 차이점은 Numerai가 각 시그널을 먼저 **자체 보유 시그널에 대해 중립화(neutralize)** 한다는 점이다.
즉, 각 시그널의 **고유한 성분(unique component)** 만을 추출하는 것이다.

채점에 사용되는 타겟(target) 역시 이러한 시그널들에 대해 중립화되어 있다.

이 과정을 통해 Numerai Signals의 핵심이 명확히 드러난다:

Numerai Signals는 단순히 주식 수익률을 예측하는 것이 아니라, Numerai가 이미 보유하지 않은 “독창적인 신호(original signal)”를 찾는 것이 목적이다.

또한, Signals Grandmasters 프로그램에는 별도의 등급 및 자격 기준이 존재한다.

자세한 내용은 [Scoring](#) 섹션을 참고하라.

스테이킹 (Staking)

Numerai Tournament와 마찬가지로, 사용자는 자신의 모델에 [NMR](#)을 스테이킹(stake) 하여 성과에 따라 NMR을 **획득하거나 소각**할 수 있다.

자세한 내용은 [Staking](#) 문서를 참고하라.

```
{% hint style="info" %}
```

스테이킹 기회는 Numerai가 제공하는 투자 계약, 증권, 파생상품, 또는 Numerai의 헤지펀드나 수익참여와 같은 제안을 의미하지 않는다.

보상은 Numerai의 재량에 따라 **비공개 블랙박스 타겟(blackbox target)** 을 기준으로 지급된다.

본질적으로 **Numerai Signals는 NMR 스테이킹을 통해 “진짜 신호(real signal)”를 검증할 수 있도록 하는 서비스다.**

Numerai는 스테이킹된 시그널 및 관련 데이터를 헤지펀드 운영에 활용한다.

이와 다른 기대를 가진 사용자는 시그널을 스테이킹하지 말아야 한다.

자세한 내용은 [이용약관\(Terms of Service\)](#)을 참고하라.

FAQ

“Ticker” 또는 “Symbol”이란 무엇인가?

주식 티커(symbol)는 거래소에서 거래되는 증권을 식별하기 위해 사용되는 문자(보통 알파벳)의 조합이다.

주식 유니버스(stock universe)란?

거래 가능한 주식 티커/심볼들의 집합을 말한다.

Numerai는 오직 이 심볼들에 대한 예측만을 원한다.

자세한 내용은 [여기](#)를 참고하라.

Numerai Tournament와 Numerai Signals의 차이는?

Signals에서는 **사용자 자신의 데이터**를 사용해야 한다.

Numerai는 피처 개발을 돕기 위해 소규모의 기본 피처 세트를 제공하지만,

Signals 참가자는 직접 피처를 수집하고 가공해 모델을 학습해야 한다.

주가 데이터나 기타 데이터를 어디서 구할 수 있나?

[이 포럼 토론](#)을 참고하라.

내 코드나 데이터가 Numerai에 노출되나?

아니다. Numerai는 사용자의 데이터 생성 또는 예측 코드를 열람하지 않는다.

Numerai는 오직 **예측 결과(predictions)** 만을 받으며,
코드를 역추적하는 것은 사실상 불가능하고, 그렇게 할 의도도 없다.

참여를 위해 반드시 NMR을 스테이킹해야 하나?

아니다. 스테이킹 없이도 예측 파일을 제출하고 성과를 확인할 수 있다.

그렇다면 직접 거래하면 되지 않나?

물론 가능하다. 그러나 Numerai는 사용자의 시그널이 이미 시장에서 널리 알려진 것인지,
즉 **차별성이 낮은 시그널인지**를 판단할 수 있다.
모두가 동일한 시그널로 거래한다면 그 신호는 가치가 낮다.
자세한 내용은 [이 Medium 글](#)을 참고하라.

내 진단(diagnostics)이 좋은지 알려줄 수 있나?

아니다. 검증 데이터에 오버피팅(overfitting)하는 것은 매우 쉽기 때문에,
진단 결과만으로 모델의 성능을 평가하는 것은 불가능하다.
스테이킹 없이 예측을 제출하고 **실시간(out-of-sample)** 성과를 꾸준히 관찰하는 것을 권장한다.
이 방법은 “페이퍼 트레이딩(paper trading)”이라고 하며,
정기적인 제출을 통해 충분한 피드백을 얻을 수 있다.

참여하려면 코딩이나 금융 지식이 필요한가?

Signals는 성공을 위해 높은 수준의 **프로그래밍 및 금융 지식**을 요구한다.
초보자는 [Numerai Tournament](#)에서 시작하는 것을 권장한다.

지원 (Support)

질문, 지원, 피드백은 [Discord](#)에서 확인하라.

데이터 (Data)

데이터 구조 (Data Structure)

Numerai Signals 데이터셋은 시간에 따른 전 세계 주식시장 정보를 담은 테이블 형식(tabular dataset)의 데이터다.

각 행(row)은 특정 시점의 특정 주식을 나타내며,
numerai_ticker 또는 composite_figi, 그리고 date로 식별된다.

date는 해당 주식의 시장 마감 데이터가 피처를 생성하는 데 사용된 날짜를 의미한다.
라운드 및 실시간 데이터의 맥락에서는 라운드가 열리기 전날을 의미한다.
target은 해당 날짜 이후 미래 수익률(예: 20일 후) 을 측정한 값이다.

피처 (Features)

Signals 데이터셋에는 다음 두 가지 주요 피처 유형이 포함되어 있다:

1. **팩터(Factors)** – 타깃이 중립화되는 요인들과 유사한 특성으로, 모델의 독창성을 평가하거나 예측값을 중립화하는 데 활용할 수 있다. 또는 자체 데이터셋에 **추가 피처**로 포함시킬 수도 있다.
2. **기본 피처(Starter features)** – 수익률 시계열(return series)로부터 생성된 **고전적 퀀트 피처(quant features)** 로, 단순하지만 학습에 유용한 시작점이 된다.

V1 피처 (V1 Features)

V1 데이터셋에 포함된 피처는 다음과 같다:

```
feature_adv_20d_factor
feature_beta_factor
feature_book_to_price_factor
feature_country
feature_dividend_yield_factor
feature_earnings_yield_factor
feature_growth_factor
feature_impact_cost_factor
feature_market_cap_factor
feature_momentum_12w_factor
feature_momentum_26w_factor
feature_momentum_52w_factor
feature_momentum_52w_less_4w_factor
feature_ppo_60d_130d_country_ranknorm
feature_ppo_60d_90d_country_ranknorm
feature_price_factor
feature_rsi_130d_country_ranknorm
feature_rsi_60d_country_ranknorm
feature_rsi_90d_country_ranknorm
feature_trix_130d_country_ranknorm
feature_trix_60d_country_ranknorm
feature_value_factor
feature_volatility_factor
```

이름에 {n}(d|w) 형식을 포함한 피처(예: feature_adv_20d_factor)는 **n일(day)** 또는 **n주(week)** 단위로 계산된 시계열(time-series) 피처를 의미한다.

country_ranknorm이 포함된 피처는
국가별로 그룹화 → 순위화(ranking) → 가우시안화(gaussianize) 과정을 거쳐 계산된다.

factor가 포함된 피처는 대부분의 타깃이 **중립화(neutralized)** 되어 있는 **리스크 팩터(risk factors)**를 의미한다.

price_factor는 국가별로 그룹화된 후 순위화된다.

- **PPO (Percentage Price Oscillator)**: 단기 및 장기 이동평균 간의 비율을 비교하는 오실레이터
- **RSI (Relative Strength Index)**: 일반적으로 과매수/과매도 상태를 측정하는 기술 지표
- **TRIX**: 세 번의 지수 이동평균을 적용한 모멘텀/반전(reversal) 지표
- **momentum_52w_less_4w**: 최근 1년간(52주)의 주가 수익률 중 **마지막 4주를 제외한** 수익률

타깃 (Targets)

이 데이터셋의 타깃은 Numerai 헤지펀드의 전략과 일치하도록 **특별히 설계된 지표**다.

Numerai의 헤지펀드는 **시장, 국가, 섹터, 팩터에 대해 중립적**이므로,

이 타깃은 쉽게 말해 시장/국가/섹터 및 알려진 요인들로 설명되지 않는 **종목 고유의 수익률**로 해석할 수 있다.

즉, 우리가 찾고자 하는 것은 바로 “**알파(alpha)**” 다.

주요 타깃 (Primary Targets)

Signals는 모델 성능을 조정(tune)하기 위한 **3가지 주요 타깃**을 제공한다.

1. target_raw_return

숙련된 데이터 사이언티스트라도 공개 데이터를 사용해 올바른 수익률 컬럼을 구축하기는 쉽지 않다.

그 이유는 주식 분할(stock split)과 배당(dividend) 같은 복잡한 요소들이 있기 때문이다.

이 타깃은 이러한 복잡성을 모두 처리하고,

그 결과를 Numerai의 표준 분포에 따라 **비닝(binning)** 한다.

→ 이 타깃은 **ICv2** 점수를 계산하는 데 사용된다.

2. target_factor_neutral

국가(country), 섹터(sector), 베타(beta), 모멘텀(momentum), 사이즈(size) 등

표준 팩터에 대해 **중립화(neutralized)** 된 타깃이다.

→ 이 타겟은 **RIC** 및 **MMC** 계산에 사용된다.

Numerai의 헤지펀드는 이러한 팩터들에 중립적이므로,

RIC은 Numerai가 실제로 Signals로부터 얻고자 하는 바를 더 잘 반영한다.

3. target_factor_feat_neutral

이 타겟은 다수의 팩터와 Numerai v4 데이터셋의 **모든 피처**에 대해 중립화되어 있다.

Signals 데이터셋의 **주요 target** 으로,

CORRv4 및 **FNCv4** 점수를 계산하는 데 사용된다.

보조 타겟 (Auxiliary Targets)

주요 타겟 외에도, 다양한 **보조 타겟(auxiliary targets)** 들이 제공된다.

이들은 모두 **주식 고유의 수익률(stock-specific returns)** 을 기반으로 하지만,

잔차화(residualization) 되는 요소(예: 시장/국가 vs 섹터/팩터)나

시간 범위(예: 20일 vs 60일) 가 다르다.

Numerai의 목표는 주요 타겟을 예측하는 것이지만,

보조 타겟을 함께 모델링하는 것이 성능 향상에 도움이 되는 경우가 많다.

때로는 보조 타겟으로 학습된 모델이 주요 타겟 모델보다 더 나은 성능을 보이기도 한다.

또한 서로 다른 타겟으로 학습된 모델들을 **앙상블(ensemble)** 하는 것도 유효한 전략이다.

참고:

일부 보조 타겟의 값은 **NaN**일 수 있다.

그러나 주요 **target**은 절대 **NaN**이 되지 않는다.

이는 특정 시점에 해당 데이터가 존재하지 않기 때문이며,

Numerai는 가짜 값을 채우지 않고 사용자가 직접 처리할 수 있도록 그대로 제공한다.

데이터 API (Data API)

Signals 데이터셋에 접근하는 가장 좋은 방법은 **데이터 API**를 사용하는 것이다.

파일 (Files)

Numerai 데이터셋은 여러 개의 파일로 구성되어 있다.

다음은 API를 통해 사용 가능한 파일 목록을 조회하고 다운로드하는 예시다:

```
from numerapi import NumerAPI
```

```

napi = NumerAPI()

DATA_VERSION = "signals/v2.0"

[f for f in napi.list_datasets() if f.startswith(DATA_VERSION)]

[f'{DATA_VERSION}/live.parquet',
 f'{DATA_VERSION}/live_example_preds.csv',
 f'{DATA_VERSION}/live_example_preds.parquet',
 f'{DATA_VERSION}/train.parquet',
 f'{DATA_VERSION}/validation.parquet',
 f'{DATA_VERSION}/validation_example_preds.csv',
 f'{DATA_VERSION}/validation_example_preds.parquet']

# Download the training data
napi.download_dataset(f'{DATA_VERSION}/train.parquet')

```

- train.parquet — 티커, 피처, 타겟이 포함된 **과거 학습 데이터**
- validation.parquet — 티커, 피처, 타겟이 포함된 **검증용 추가 과거 데이터**
- live.parquet — **현재 라운드의 최신 티커 유니버스 및 실시간 피처** (타겟 없음)
- live_example_preds.parquet — 예시 모델의 **최신 실시간 예측값**
- validation_example_preds.parquet — 예시 모델의 **검증용 예측값**

모델 (Models)

Numerai 및 커뮤니티에서 제작한 다양한 튜토리얼 모델이 제공됩니다:

- Numerai의 기본 예시 모델은 [GitHub](#)에서 확인할 수 있다.
- [Jason Rosenfeld \(jrAI\)](#)는 Yahoo Finance의 무료 데이터를 활용해 데이터셋을 구축하고 간단한 모델을 학습하는 **Numerai Signals용 종단 간 노트북**을 만들었다.
Google Colab에서 [노트북](#)을 확인할 수 있으며,
[관련 영상](#)도 제공된다.
- [Surajp](#)는 위 예시 및 노트북에서 영감을 받아
Signals에 대해 [이 글](#)을 작성했다.
- 데이터 소스 목록은 [Numerai 포럼](#)에서 확인할 수 있다.

제출 (Submissions)

시그널 제출이란? (What is a Signal submission?)

시그널을 생성하기 위한 원본 데이터는 매우 다양할 수 있다.

예: 감사받은 재무제표(audited financials), 뉴스 기사(news articles), 주차장 위성사진(images of parking lots) 등.

그러나 모든 Signals 제출물은 기본 형식을 동일하게 따라야 한다.

즉, 각 주식 티커(stock ticker)에 대해 0과 1 사이의 수치 값을 갖는 리스트 형태여야 한다.

제출 파일의 주식 티커 목록은 Numerai Signals 주식시장 유니버스(Numerai Signals stock market universe)에 의해 정의된다.

이 유니버스는 전 세계 상위 약 5,000개 주식을 포함하며, 매일 업데이트된다.

일반적으로 거래량이 적은 몇몇 주식만이 하루 동안 유입되거나 제외된다.

최신 유니버스는 [live.parquet](#) 파일을 다운로드하여 확인할 수 있다.

시그널을 제출할 때는 최소 두 개의 컬럼이 필요하다:

- `cusip`, `sedol`, `bloomberg_ticker`, `composite_figi`, 또는 `numerai_ticker` — 헤더에 명시된 티커 형식에 해당하는 **유효한 티커 값**이어야 한다.
- `signal` — 0과 1 사이(0과 1은 포함되지 않음)의 **실수 값**이어야 한다.

또한 live 제출이 유효하기 위해서는 다음 조건을 만족해야 한다:

- 최소 100개 이상의 유효한 티커(0~1 값)가 포함되어야 한다.
- 동일한 티커가 현재 live 기간 내에 두 번 이상 등장할 수 없다.

제출 시기 (When to submit)

제출 창(submission window)이 언제 열리고 닫히는지 이해하려면

[Numerai Tournament 제출 문서](#)를 참고하라.

자동화 (How to automate)

Signals 제출 프로세스는 API를 사용해 자동화할 수 있다. 예측 결과를 포함한 Parquet 또는 CSV 파일을 생성한 후, 다음 예시 코드와 같이 제출을 자동화할 수 있다:

```
from numerapi import SignalsAPI

# Authenticate
sapi = SignalsAPI("[your api public id]", "[your api secret key]")
sapi.upload_predictions("[path to your submission]", model_id="your-model-id")
```

- 다음 중 아무 것이나 사용할 수 있다
 - [Numerapi](#) (공식 Python 클라이언트)
 - [RNumerai](#) (비공식 R 클라이언트)
 - 다른 언어용 원시 [GraphQL API](#)
- 오픈소스 클라우드 자동화 도구 [Numerai Compute](#)

채점 (Scoring)

현재 보상에 사용되는 주요 점수는 두 가지다

- [Feature Neutral Correlation](#) (FNCv4): 중립화한 예측과 타겟의 상관관계
 - 이에 사용되는 타겟은 target_factor_feat_neutral_20 (일반 피처와 팩터에 대해 중립화되어, “잔차 수익률(residual return)”을 산출)
- [Meta Model Contribution](#) (MMC): 메타 모델에 대한 예측의 기여도

Numerai는 2025년 9월 2일 이후(해당 날짜 포함)의 라운드부터 다른 점수로 보상을 전환할 예정이다:

- Alpha: chili 타겟으로 가중한 중립 가중 예측값
- Meta Portfolio Contribution (MPC): 메타 포트폴리오에 대한 예측의 기여도

또한 보상에는 사용되지 않는 정보성 점수들도 있다:

- [Correlation](#) (CORRv4): 예측과 타겟의 상관관계
- Information Coefficient (ICv2): 원시 수익률과의 상관관계
- Residual Information Coefficient (RIC): 일반 팩터에 대해 중립화된 잔차 수익률과의 상관

자세한 설명의 전체 목록은 [정의 문서](#)를 참고하라.

중립화 (Neutralization)

시그널이나 타겟이 기존 시그널 집합과의 상관이 0이면 “중립(neutral)”로 간주된다. 중립화의 목적은 기존 시그널에 이미 존재하지 않는 **원천적/직교 성분**을 고립시키는 데 있다.



```
{% hint style="warning" %}
```

몇 가지 널리 알려진 시그널의 단순 선형결합을 제출하면, 중립화 이후 남는 직교 성분이 거의 없거나 아예 없을 수 있다.

```
{% endhint %}
```

Numerai는 Barra 팩터(예: 사이즈, 밸류, 모멘텀 등), 국가/섹터 리스크 팩터, 맞춤형 주식 피처 등 다양한 기존 시그널을 보유하고 있다. 이러한 기존 시그널이 모두 사용자에게 제공되는 것은 아니므로, 이 과정은 다소 “블랙박스”처럼 느껴질 수 있다.

채점 전에 시그널을 중립화함으로써 Numerai는 중립화된 타겟과의 정렬을 달성하며, 중립화에 사용된 데이터를 공개하지 않고도 타겟 대비 성능을 개선할 수 있다. 예를 들어, 사용자의 시그널이 국가 리스크에 대해 중립화되어 있지 않다면, Numerai Signals는 채점 전에 국가 리스크에 대해 시그널을 중립화한다. 이는 사용자가 국가 리스크 중립화에 신경 쓰지 않고 **독창적 시그널** 제작에 집중할 수 있게 한다.

어떤 시그널은 단독으로 볼 때 예측력이 강할 수 있지만, 이러한 중립화 때문에 Numerai Signals에서 점수가 낮게 나올 수도 있다. 이는 Signals의 핵심적, 독특한 측면을 강조한다: **Numerai Signals는 단순히 주가 수익률을 예측하는 것이 아니라, Numerai가 아직 보유하지 않은 독창적인 시그널을 찾는 것이다.**

타겟 (The Target)

시그널은 기존 시그널에 대해 중립화된 커스텀 블랙박스 타겟에 대해 평가된다.

데이터셋에는 [20D2L](#) 및 [60D2L](#) 타겟이 제공된다. 더 짧은 시간 구간은 사용하지 않는데, 이는 짧은 구간에서만 작동하는 시그널은 대형 헤지펀드가 실제로 거래하기 거의 불가능하기 때문이다. 예를 들어, 어떤 시그널이 1시간 수익률을 정확히 예측할 수 있더라도, 헤지펀드가 그 포지션에 완전히 진입하는 데 24시간이 걸린다면 실사용 가치는 낮다. 대형 헤지펀드에 가장 유용한 시그널은 **장기 시간 구간에서 예측력**을 가지며, 이를 낮은 **알파 감쇠(low alpha decay)**라고도 한다.

그랜드마스터 (Grandmasters)

Grandmasters 는 1위를 차지한다

Masters 는 상위 10위에 든다

Experts 는 상위 25%에 든다

Researchers 는 상위 50%에 든다

Contributors 는 상위 75%에 든다

Apprentices 는 하위 25%에 든다

Novices 는 아직 20회의 유효 제출을 완료하지 못했다

캐논 점수 (Canon Scores)

Signals 토너먼트의 맥락에서 Canonical Scores(“Canon Scores”)는 특히 중요하다. 예를 들어, 보상 지표인 FNC 점수는 업데이트를 거쳤다. 초기에는 라운드 498까지 보상 점수가 ‘CORR20’이었고, 라운드 499부터 ‘FNCv4’로 발전했다. ‘Canon FNC’ 점수는 이러한 변화를 통합하여, **라운드 498 이하에서는 ‘CORR20’, 그 이후 라운드에서는 ‘FNCv4’** 를 사용하는 **통합 점수**로 정의된다.

진단 (Diagnostics)

사용자는 시그널의 과거 진단 결과를 이용해 성과를 점검하고, 향후 중립화가 시그널에 미칠 영향을 추정할 수 있다. 과거 기간에서 점수가 높더라도, 현재나 미래의 어떤 라운드에서도 높은 점수를 보장하지는 않는다는 점에 유의해야 한다.

진단 도구는 [scores](#) 페이지에서 모델 옆의 비커(beaker) 아이콘을 눌러 열 수 있다. 과거 `validation` 기간에 대한 시그널을 업로드하면, 성과, 리스크, 잠재 수익 등을 포함한 `validation` 메트릭을 계산한다. `validation` 기간은 20130104에 시작하여 `validation` 데이터의 최신 날짜에서 종료된다.

`validation` 기간에 대한 업로드에는 한 개의 추가 컬럼이 필요하다:

- `date` 컬럼 — 과거 데이터는 주간(weekly)이며, 진단 도구는 특정 주에 대한 예측이 해당 주의 **마지막 금요일** 종가 데이터를 사용해 만들어졌다고 가정한다

numerai_ticker	date	signal
000060 KR	20130104	0.601189
000080 KR	20130104	0.911796
000100 KR	20130104	0.637412
000120 KR	20130104	0.441297
000150 KR	20130104	0.434370
...	...	...
ZUO US	20230825	0.583844
ZURN CH	20230825	0.686535
ZWS US	20230825	0.461798
ZYME US	20230825	0.480338
ZZZ CA	20230825	0.027202

업로드가 유효성 검사를 통과하면 진단이 실행되기 시작한다. 이는 제출물에 포함된 주차(weeks)와 티커 수에 따라 보통 5–10분이 소요된다.

이 진단은 스테이킹할 가치가 있는지 추정하는 데 도움을 주는 가이드로 활용된다. 다만, 과거 validation 기간에서 좋은 진단 결과를 보였더라도, 현재 또는 미래의 live 기간에서 좋은 점수를 보장하지는 않는다는 점에 유의해야 한다.

```
{% hint style="warning" %}
```

이 과거 평가 도구를 반복적으로 사용하면 금방 오버피팅에 빠질 수 있다. 진단은 시그널 제작 과정의 **최종 점검** 정도로만 사용하라.

```
{% endhint %}
```

Churn과 Turnover란 무엇인가? (What is Churn and Turnover?)

“Churn”은 시그널이 시간에 따라 **얼마나 변하는지**를 나타내는 통계량이다. 유사하게, “turnover”는 포트폴리오가 시간에 따라 **얼마나 바뀌는지**를 나타낸다. Signals에서 우리가 사용하는 churn 및 turnover 계산 코드는 오픈소스로 공개했다. 코드는 [여기](#)에서 확인할 수 있다. 간단히 요약하면:

$$\text{churn}(t_0, t_1) = 1 - \text{correlation}(s(t_0), s(t_1))$$

$$\text{turnover}(t_0, t_1) = (p(t_0) - p(t_1)).\text{abs}().\text{sum}() / 2$$

여기서 $s(t)$ 는 시각 t 에서의 시그널, $p(t)$ 는 시각 t 에서의 포트폴리오를 의미한다.

왜 churn 또는 turnover를 계산하는가?

Signals 제출물이 높은 churn/turnover를 보이면, Numerai는 그 시그널을 거래할 수 없다. 원래 Numerai 토너먼트 데이터로 구축된 많은 모델들은 자연스럽게 낮은 churn/turnover를 보이지만, Signals 모델은 대체로 높은 churn/turnover를 보이는 경향이 있다.

이는 Signals 메타 모델의 churn에 부정적 영향을 미친다는 것을 알고 있다. 개별 Signals 모델의 평균 churn이 Signals 메타 모델의 churn과 높은 상관을 가지기 때문이다. 따라서 Numerai는 churn과 turnover가 높은 Signals 모델을 허용할 수 없다.

Churn 및 Turnover 임계값 (Thresholds)

이전 주에 제출을 하지 않은 모든 모델의 스테이크는 0으로 설정된다. 매주 제출하지 않는 모델은 메타 모델의 churn과 포트폴리오 turnover를 자연스럽게 높이기 때문이다.

모델이 지난 주 내 제출 이력이 있다면, 새 제출을 업로드할 때 **이전 주부터 최대 5주 전까지의 제출**을 기준으로 최대 churn 및 turnover를 계산한다. 현재 업로드 기간을 시각 t 라고 하면, 최대 churn/turnover는 다음과 같다:

$$\text{max_churn} = \max([\text{churn}(t, t-1), \text{churn}(t, t-2), \dots, \text{churn}(t, t-5)])$$

$$\text{max_turnover} = \max([\text{turnover}(t, t-1), \text{turnover}(t, t-2), \dots, \text{turnover}(t, t-5)])$$

max_churn $\geq 15\%$ 또는 **max_turnover** $\geq 20\%$ 이면 해당 제출물의 스테이크는 0으로 설정된다.

정의 (Definitions)

아래의 기본 함수들을 이해하려면 [메인 정의 문서](#)를 참고하라.

타겟 (Targets)

- target_raw_return_20/60
 - 타임라인: 20D2L / 60D2L
 - bins=5, uniformity=10%, 40%, 50%
 - 종립화 대상: 없음
- target_factor_neutral_20/60
 - 타임라인: 20D2L / 60D2L
 - bins=5, uniformity=10%, 40%, 50%
 - 종립화 대상: 표준 팩터(Standard Factors)
- target_factor_feat_neutral_20/60
 - 타임라인: 20D2L / 60D2L
 - bins=5, uniformity=10%, 40%, 50%
 - 종립화 대상: 표준 팩터 및 기타 비공개 피쳐
- target_chili_60
 - 타임라인: 60D2L
 - bins=5, uniformity=10%, 40%, 50%
 - 종립화 대상: 전부(everything) 및 Numerai 독점 리스크 모델
 - 기타 조정: ADV 가중 및 레벨별 익스포저 제약

메타 모델 (Meta Models)

- Signals 제출물은 먼저 다음과 같이 정제된다:
 - 각 제출물에 tie-kept-rank 적용
 - NaN을 0.5로 대체
 - 다시 tie-kept-rank 적용
 - 각 제출물 가우시안화
- 최소 스테이크 조건의 Signals 스테이크 가중 메타 모델 (SSWMM)
 - 정제된 Signals 제출물의 스테이크 가중 평균
- 최소 스테이크 조건의 Signals 단순 가중 메타 모델 (SNWMM)
 - 정제된 Signals 제출물의 단순 평균
- 최소 10 NMR 스테이크 조건의 Signals 단순 가중 메타 모델 (SNWMMmin10)
 - 최소 10 NMR 스테이크가 있는 정제된 제출물들의 단순 평균

메타 포트폴리오 (Meta Portfolio)

- 중립 가중(neutral weights)
 - 제출물에 중립화 행렬과 샘플 가중 벡터를 적용
- 스테이크 가중 포트폴리오 (SWP)
 - Signals 내 모든 중립 가중의 스테이크 가중 평균

점수 (Scores)

- 채점에 사용되기 전 제출물은 정제된다:
 - 유효하지 않은 티커 제거
 - 제출물에 tie-kept rank 적용
 - NaN을 0.5로 대체
- ALPHA
 - target_chili_60과 중립 가중의 내적(dot product)
 - 타임라인: 60 채점일 + 수익률 지연 2일(+ 데이터 지연 2일)
- MPC - Meta Portfolio Contribution
 - SWP의 그래디언트에 target_chili_60을 곱하고 스테이크에 대해 미분
 - 타임라인: 60 채점일 + 수익률 지연 2일(+ 데이터 지연 2일)
- MMC - Meta Model Contribution
 - 제출물, SNWMMmin10, target_factor_neutral_20의 상관 기여도
 - 타임라인: 20 채점일 + 수익률 지연 2일(+ 데이터 지연 2일)
- CORRV4 - Correlation v4
 - 제출물과 target_factor_feat_neutral_20의 numerai corr
 - 타임라인: 20 채점일 + 수익률 지연 2일(+ 데이터 지연 2일)
- ICV2 - Information Coefficient V2
 - 비닝된 수익률과 제출물 간 numerai corr
 - 타임라인: 20 채점일 + 수익률 지연 2일(+ 데이터 지연 2일)
- RIC - Residual Information Coefficient
 - target_factor_neutral_20과 제출물 간 numerai corr
 - 타임라인: 20 채점일 + 수익률 지연 2일(+ 데이터 지연 2일)
- FNCV4 - Feature Neutral Correlation v4
 - 제출물에 tie-kept-rank → 가우시안화 → 중립화 적용
 - target_factor_feat_neutral_20과 제출물 간 tie-broken-rank 상관
 - 중립화 대상: 표준 팩터 및 V4 Medium Safe Features

- 타임라인: 20 채점일 + 수익률 지연 2일(+ 데이터 지연 2일)
- CWSNMM - Corr w/ Signals Naive Meta Model
 - $s' = \text{tie-kept rank} \rightarrow$ 가우시안화 $\rightarrow$ 제출물 s 에 pow 1.5 적용
 - s' 와 SNWMM 간 피어슨 상관 계산
 - 타임라인: 데이터 지연 4일 / 수익률 비의존
- MCWSM - Max Corr w/ Signals Models
 - 제출물과 다른 Signals 제출물 간 **최대** 피어슨 상관
 - 같은 라운드 내 제출들과만 비교
 - 타임라인: 데이터 지연 4일 / 수익률 비의존
- APCWSM - Average Pairwise Corr w/ Signals Models
 - 제출물과 다른 각 Signals 제출물 간 **평균** 피어슨 상관
 - 같은 라운드 내 제출들과만 비교
 - 타임라인: 데이터 지연 4일 / 수익률 비의존

Alpha

CORR, FNC, MMC는 Signals 사용자에게 보상을 지급하기에 **불완전한 지표**다. 이는 사용자가 이러한 점수에서 좋은 성과를 보이더라도 **백테스트 성과가 부진**할 수 있기 때문이다. Numerai Signals 토너먼트를 Numerai 헤지펀드와 더 잘 정렬시키려면, 제출물의 성과가 **백테스트 성과와 정합**되어야 한다.

Numerai 리서치 팀은 이를 가능하게 하는 두 가지 데이터를 고안했다:

- **Neutralizers Matrix**: 중립화 시그널을 생성하는 데 사용되는 200열의 축약 행렬
- **Sample Weights Vector**: adv와 같은 실거래 제약을 반영하여 시그널을 표본추출해 주식 가중을 산출하는 가중 벡터

이를 사용해, 시그널로부터 다음과 같이 “**중립화된 가중(neutralized weights)**”을 만들 수 있다:

다음 데이터가 주어졌다고 하자:

- 시그널 s (NaN은 0.5로 채움)
- 중립화 행렬 N
- 가중 벡터 v

먼저 정규화된 시그널 s' 를 생성한다:

```
s` = normalize(rank(s))^1.5
```

그다음 중립화된 예측을 생성하고 이를 샘플링한다:

```
neutral_preds = s` - (N @ (N.T @ (v * s`)))  
neutral_weights = neutral_preds * v
```

마지막으로, 60D2L 타킷 `chili` 를 사용해 최종 `alpha` 값을 도출한다:

```
alpha = neutral_weights @ target_chili_60
```

이 값은 시그널이 백테스트에서 어떻게 작동할지에 대한 통찰을 제공하며, Signals 토너먼트를 헤지펀드와 더 긴밀하게 정렬시킨다.

메타 포트폴리오 기여도 (Meta Portfolio Contribution, MPC)

Alpha가 도입됨에 따라, Signals에는 MMC와 유사하지만 Alpha에서 사용하는 것과 동일한 **중립화 행렬(neutralizers matrix)**과 **샘플 가중 벡터(sample weights vector)**를 활용하는 지표가 필요했다. 메타 모델의 아날로그로는 **스테이크 가중 포트폴리오(SWP, Stake-Weighted Portfolio)**를 생각할 수 있다. 이 경우, **메타 포트폴리오 기여도(MPC)**는 특정 시그널이 SWP의 Alpha를 얼마나 향상시키는지 평가한다.

개별 사용자의 기여도를 구하기 위해, SWP에서 해당 사용자의 스테이크를 소폭 증가시킨 뒤 SWP의 Alpha 변화량을 측정할 수 있다. 본질적으로 이는 **사용자 스테이크에 대한 SWP Alpha의 그래디언트**에 해당한다.

다음 데이터가 주어진다:

- 시그널 행렬 S (NaN은 0.5로 채움)
- 스테이크 가중치 p
- 중립화 행렬 N
- 가중 벡터 v

먼저 시그널별 **중립 가중(neutral weights)**을 생성한다:

```
s` = normalize(rank(s))^1.5  
neutral_preds = s` - (N @ (N.T @ (v * s`)))  
neutral_weights = neutral_preds * v
```

그다음 이를 스테이크 가중 평균하여 **SWP** 를 만들고, 전체 가중을 정규화한다:

```
SWP = (p * user_stakes).sum()  
SWP = SWP - SWP.mean()  
SWP = SWP / sum(abs(SWP))
```

이제 SWP의 Alpha를 계산할 수 있다:

```
SWP_Alpha = SWP * target
```

마지막으로, **MPC**는 스테이크에 대한 **SWP Alpha**의 그래디언트다:

```
MPC = gradient(SWP_Alpha, p)
```

MPC를 기준으로 보상이 지급될 때, 데이터 사이언티스트는 Numerai Signals 토너먼트를 Numerai 헤지펀드에 유용하게 만드는 방향으로 기여하는 시그널에 스테이킹하도록 동기 부여된다. MPC 코드가 궁금하다면 [numerai-tools 저장소](#)를 확인하라.

스테이킹 (Staking)

시그널의 [churn 및 turnover](#)가 충분히 낮다면, 모델에 [NMR](#)을 **스테이킹(stake)** 하여 점수에 따라 NMR을 획득하거나 소각할 수 있다. 스테이킹이란 사용자의 NMR을 잠그고, 잠긴 NMR에 대해 Numerai가 보상을 추가하거나 소각할 수 있도록 허용하는 것을 의미한다. 스테이킹과 보상에 대한 일반 규칙은 [메인 스테이킹 문서](#)를 참고하라.

시그널을 스테이킹할 기회는 Numerai가 제공하는 **투자 계약, 증권, 금융자산 수익률 기반 스왑**, Numerai 헤지펀드의 지분 또는 Numerai 자체(또는 Numerai의 수수료 수익)에 대한 참여 제안을 의미하지 않는다. 보상은 사용자에게 공개되지 않는 **블랙박스 타깃**을 기준으로 Numerai의 재량에 따라 지급된다. 근본적으로 **Numerai Signals**는 **NMR 스테이킹을 통해 “진짜(real)” 시그널을 검증**하고, 사용자가 자신의 시그널 가치를 평가할 수 있도록 하는 서비스다. 그 대가로 Numerai는 스테이킹된 시그널과 관련 데이터를 Numerai 헤지펀드에서 활용한다. 이와 다른 기대를 가진 사용자는 시그널을 스테이킹하지 말아야 한다.

보상 (Payouts)

보상은 사용자의 스테이크 값과 점수의 함수다. 보상 방식 전반에 대해서는 [메인 스테이킹 문서](#)를 검토하라. Signals의 구체적인 보상 함수는 다음과 같다:

```
payout = stake * clip(payout_factor * (fncv4 * 1 + mmc * 2), -0.05,  
0.05)
```

2025년 9월 2일 이후 시작되는 라운드부터는 **보상 함수(payout function)**가 변경된다:

```
payout = stake * clip(payout_factor * (alpha * 0.3 + mpc * 0.8), -0.017, 0.017)
```

payout_factor는 전체 스테이킹된 NMR 총량이 stake_cap_threshold를 초과함에 따라 **로그 함수(logarithmic)** 형태로 감소한다.
stake_cap_threshold의 값은 [여기](#)에서 확인할 수 있다.

Signals + QuantConnect

QuantConnect는 2012년에 설립되어 전 세계 퀀트들에게 최고의 알고리즘 트레이딩 기술을 제공하는 것을 목표로 한다.

폐쇄적인 금융 산업을 혁신하기 위해 **오픈소스 중심의 알고리즘 트레이딩 플랫폼**을 구축했으며, 현재 매월 50,000명 이상의 퀀트들이 **Equities, Options, Futures, Crypto, Forex, CFD** 등 다양한 자산군을 대상으로 백테스트, 리서치, 트레이딩을 수행하고 있다.

Numerai Signals는 사용자가 어떤 데이터셋이든 활용해 거래 시그널을 생성할 수 있게 해준다.

QuantConnect를 활용하면 [다양한 대체 데이터 라이브러리](#)나

직접 만든 [커스텀 데이터셋](#)을 이용해

고유한 트레이딩 시그널을 만들 수 있다.

QuantConnect 팀은 데이터가 올바르게 처리되고 제때 전달되도록 보장하며, API를 통해 자산 유니버스 선택, 트레이딩 로직 정의, Numerai Signals 제출 예약 등을 손쉽게 구현할 수 있다.

QuantConnect에서 실시간 알고리즘을 배포하면, 엔지니어 팀이 인프라를 모니터링해 사용자는 단순히 Numerai Signals 제출을 자동화하고 리서치에 집중할 수 있다.

요구사항 (Requirements)

QuantConnect의 **공동 서버(co-located servers)**에서 실시간 알고리즘을 실행하려면, [organization](#)에 속해 있어야 하며 유휴 상태의 [live trading node](#)가 필요하다.

- **최소 비용:**
 - Quant Researcher 조직 좌석: 월 \$8

- 실시간 트레이딩 노드: 월 \$24

Numerai Signals 연결 (Connecting to Numerai Signals)

QuantConnect 알고리즘에서 Numerai로 시그널을 내보내려면, 초기화 단계([initialization](#))에서

Numerai Signal Export Provider를 추가해야 한다.

```
self.SignalExport.AddSignalExportProviders(NumeraiSignalExport(publicId, secretId, modelId, fileName))
```

NumeraiSignalExport 생성자는 다음 인자를 받는다:

인자	데이터 타입	설명	기본값
publicId	string	Numerai API 키	—
secretId	string	Numerai API 시크릿 키	—
modelId	string	Numerai 모델 ID	—
fileName	string	시그널 파일 이름. 동일 이름을 사용하면 기존 파일이 덮어쓰워짐	"predictions.csv"

유니버스 선택 (Universe Selection)

Numerai Signals의 주식시장 유니버스는 전 세계 약 **상위 5,000개 주식**을 포함하지만, QuantConnect 클라우드 플랫폼은 현재 **미국 주식(US Equities)** 만 실시간 트레이딩을 지원한다. Numerai Signals 유니버스와 가장 유사한 것은 **CRSP US Total Market Index** 로, 뉴욕증권거래소(NYSE)와 나스닥(Nasdaq)에서 정기적으로 거래되는 **투자 가능한 미국 주식시장 전체(약 100%)** 를 대표한다.

이 인덱스가 Numerai Signals의 전체 유니버스를 포함하지는 않지만, 모든 주식에 대한 시그널을 제출할 필요는 없다.

CRSP US Total Market Index 구성 종목을 얻으려면,
ETF 구성 종목 유니버스로 **Vanguard Total Stock Market ETF (VTI)** 를 추가한다.

```
self.etf_symbol = self.AddEquity("VTI").Symbol
self.AddUniverse(self.Universe.ETF(self.etf_symbol))
```

유니버스 구성 종목이 변경되면,
OnSecuritiesChanged 이벤트 핸들러를 통해 알고리즘에 알림이 전송된다.
클래스에 securities 리스트를 추가하여 이를 관리하고,
OnSecuritiesChanged 이벤트 핸들러에서 해당 리스트를 업데이트하도록 정의할 수 있다.

```
def OnSecuritiesChanged(self, changes: SecurityChanges) -> None:
    for security in changes.RemovedSecurities:
        if security in self.securities:
            self.securities.remove(security)

    self.securities.extend([security for security in
changes.AddedSecurities if security.Symbol != self.etf_symbol])
```

포트폴리오 구성 (Portfolio Construction)

Numerai Signals에 시그널을 제출할 때는 다음 규칙을 따라야 한다:

- 모든 시그널 값은 **0과 1 사이(0, 1 제외)** 여야 한다.
- 제출마다 최소 **10개 이상의 시그널**을 포함해야 한다.
- 유니버스 내 모든 종목의 시그널 값이 동일해서는 안 되며, 최소 한 개 이상은 **고유한 값**이어야 한다.

시그널 생성을 위해 PortfolioTarget 객체를 생성한다.

```
targets = [PortfolioTarget(symbol, weight) for symbol, weight in
weight_by_symbol.items()]
```

Algorithm Framework 설계를 사용하는 경우,
여러 사전 구축된 포트폴리오 구성 모델을 활용할 수 있다.
단, 모든 종목에 동일 가중을 부여하는 **Equal Weighting 모델**은 사용할 수 없다.
대부분의 경우 **Insight Weighting 모델**을 사용하는 것이 권장되며,
이 모델은 생성된 시그널의 강도에 따라 자산 가중을 부여한다.

실제로 거래를 실행할 필요는 없지만,
알고리즘의 활동을 시각화하기 위해 거래를 포함시킬 수도 있다.
Numerai에 시그널을 전송하려면 SetTargetPortfolio 메서드를 호출하면 된다.
이 메서드는 Numerai로 시그널이 성공적으로 전송되었는지를 나타내는 **불리언(Boolean)** 값을 반환한

다.

```
success = self.SignalExport.SetTargetPortfolio(targets)
```

시그널 스케줄링 (Scheduling Signals)

Numerai Signals는 매주 화, 수, 목, 금, 토요일에 새로운 라운드를 연다.

- **토요일 라운드**: 18:00 UTC 오픈 → 월요일 14:30 UTC 마감
- **평일 라운드**: 13:00 UTC 오픈 → 1시간 동안 제출 가능

자세한 라운드 일정은 [Rounds 문서](#)를 참고하라.

제출을 예약하려면 [Scheduled Event](#)를 사용한다.

```
self.Schedule.On(  
    self.DateRules.EveryDay(self.etf_symbol),  
    self.TimeRules.At(13, 0, TimeZones.Utc),  
    self.submit_signals)
```

이전 예시에서는 submit_signals 메서드가 매 거래일 **13:00 UTC**에 자동 실행된다.

실시간 알고리즘 배포 (Deploying Live Algorithms)

Numerai Signals에 시그널을 제출하는 QuantConnect 실시간 알고리즘을 배포하려면 다음 단계를 따른다:

1. [새 프로젝트 생성](#)
2. [Initialize 메서드](#)에 **미국 주식 유니버스** 추가

```
self.etf_symbol = self.AddEquity("VTI").Symbol  
self.AddUniverse(self.Universe.ETF(self.etf_symbol))
```

3. Initialize 메서드에 **Numerai Signal Export Provider** 추가

```
self.SignalExport.AddSignalExportProviders(NumeraiSignalExport(publicId,  
secretId, modelId, fileName))
```

4. Initialize 메서드에 각 라운드별 시그널 제출용 **Scheduled Event** 추가

```
self.Schedule.On(
    self.DateRules.EveryDay(self.etf_symbol),
    self.TimeRules.At(13, 0, TimeZones.Utc),
    self.submit_signals)
```

5. 클래스 멤버로 유니버스 내 종목을 추적할 **securities** 리스트 정의

```
securities = []
```

6. 유니버스 변경 시 OnSecuritiesChanged 이벤트 핸들러를 정의해 securities 리스트 업데이트

```
def OnSecuritiesChanged(self, changes: SecurityChanges) -> None:
    for security in changes.RemovedSecurities:
        if security in self.securities:
            self.securities.remove(security)

    self.securities.extend([security for security in
        changes.AddedSecurities if security.Symbol != self.etf_symbol])
```

7. Scheduled Event에서 시그널 생성 및 전송 로직 정의

```
def submit_signals(self):
    # Select the subset of ETF constituents we can trade
    symbols = [security.Symbol for security in self.securities if
        security.HasData]
    if len(symbols) == 0:
        return

    # Get historical data
    # close_prices = self.History(symbols, 22,
    Resolution.Daily).close.unstack(0)

    # Create portfolio targets
    # weight_by_symbol = {} # Add your logic here
    targets = [PortfolioTarget(symbol, weight) for symbol, weight in
        weight_by_symbol.items()]

    # (Optional) Place trades
    self.SetHoldings(targets)

    # Send signals to Numerai
    success = self.SignalExport.SetTargetPortfolio(targets)
```

```
if not success:
    self.Debug(f"Couldn't send targets at {self.Time}")
```

과거 주가나 대체 데이터를 요청하려면 [History Requests](#)를 참고하라.

8. 웹 IDE 상단의 **Backtest** 아이콘을 클릭

QuantConnect는 백테스트 중 Numerai로 시그널을 전송하지 않지만, 배포 전 알고리즘 코드에 오류가 있는지 빠르게 확인할 수 있다.

9. 알고리즘을 실시간 거래로 배포

실제 자금 거래를 원하지 않으면,

[Paper Trading 브로커리지](#)로 배포할 수 있다.

실제 거래와 Numerai 제출을 동시에 수행하려면,

[지원 브로커리지 목록](#)을 참고하라.

다음 단계 (Next Steps)

QuantConnect를 사용해 Numerai Signals 제출을 자동화하려면,

[Numerai Signals Overview 문서](#)를 검토하고

[QuantConnect 계정 생성](#)을 진행하라.

데모 알고리즘 (Demonstration Algorithm)

Numerai Signals에 시그널을 제출하는 전체 예시 알고리즘은

LEAN GitHub 저장소의

[NumeraiSignalExportDemonstrationAlgorithm](#)에서 확인할 수 있다.

Numerai Crypto

개요 (Overview)

소개 (Introduction)

[Numerai Signals](#)와 유사하게, [Numerai Crypto](#)는 사용자가 **자신만의 데이터**, 즉 **고유한 시그널**을 가져오도록 요구한다. 암호화폐 시장에서의 시그널(“CryptoSignal”)은 정보 피드(feed of information)로, 트레이더가 활용할 수 있는 토큰에 대한 수치적 데이터다. 이러한 정보 피드는 암호화폐 시장을 모델링하고 **Numerai Crypto 메타 모델**을 구성하는 데 사용되며, 이 메타 모델은 여러분에게 **무료로** 제공된다.

크립토 시장 시그널의 예시는 다음과 같다:

- 기술적 신호(Technical signals) (MACD, RSI, MFI)
- 대체 데이터 신호(Alternative data signals) (신용카드 거래, 위성 이미지, 소셜 미디어 감성)
- 블렌디드 신호(Blended signals) (Barra 리스크 요인, Fama-French 요인)

데이터 제공자는 고유한 **피처(features)** 를 시그널로 직접 제출할 수 있고, 데이터 사이언티스트는 고유 데이터를 모델링해 **예측값(predictions)** 을 시그널로 제출할 수 있다. 제출된 시그널은 우리의 타깃 및 다른 제출 시그널과 비교되어 채점된다. 또한 NMR 암호화폐로 시그널을 **스테이킹**하여 성과에 따라 NMR을 획득(earn)하거나 소각(burn)할 수 있다.

데이터 (Data)

Numerai Data API로 시작할 수 있다:

고품질 시그널을 만들려면, 독자적이고 고유한 크립토 시장 데이터를 반드시 확보해야 한다. 시작을 위해 사용할 수 있는 데이터 제공업체로는 Messari와 CoinMarketCap 등이 있다.

만약 크립토 시장 데이터를 보유하고 있지 않지만 Numerai에 관심이 있다면, 우리의 데이터를 사용해 주식 시장을 예측하는 Numerai Tournament를 대신 시도해 보라.

모델링 (Modeling)

데이터 모델링을 어디서 시작해야 할지 모르겠다면, Numerai Tournament에서 제공하는 기본 모델링 을 먼저 학습하는 것을 강력히 추천한다. 모델링에 자신이 생기면 그 기술은 자연스럽게 Numerai Crypto 에도 적용된다. 다음은 트리 기반 모델의 간단한 예시다:

제출 (Submissions)

Numerai Crypto의 제출 방식은 Numerai Signals와 매우 유사하지만, 주식 티커 대신 **토큰 심볼(symbol)** 을 사용한다. 제출물은 **수익률(returns)** 을 예측하는 각 심볼과 그에 대응하는 하나의 수치 값 (0~1) 의 목록이어야 한다:

symbol	signal
BTC	0.529851
ETH	0.376866
SOL	0.272388
NMR	0.929104

제출에 포함되는 심볼 목록은 **Numerai Crypto 토큰 유니버스**로 정의되며, 각 수치 값은 **0과 1** 사이여야 한다.

다음은 Python으로 실시간 예측을 생성 및 업로드하는 예시다:

제출에 대한 자세한 내용은 [여기](#)를 참고하라.

채점 (Scoring)

채점은 [Numerai Tournament](#)와 매우 유사하다.

Numerai Crypto는 단순히 토큰 수익률을 예측하는 것이 아니라, 다른 모델에 아직 존재하지 않는 **독창적 시그널**을 찾는 데 목적이 있다.

자세한 내용은 [Scoring](#) 섹션을 참고하라.

스테이킹 (Staking)

Numerai Tournament와 마찬가지로, 성과에 따라 NMR을 획득하거나 소각할 수 있도록 모델에 선택적으로 stake [NMR](#)할 수 있다. 자세한 내용은 [Staking 문서](#)를 참고하라.

시그널을 스테이킹할 기회는 Numerai가 제공하는 투자 계약, 증권, 금융자산 수익률 기반 스왑, Numerai의 헤지펀드 지분 또는 Numerai 자체(혹은 수수료 수익)에 대한 참여 제안을 의미하지 않는다. Numerai의 헤지펀드는 Numerai Crypto와 어떤 관련도 없으며, 암호화폐를 거래하지 않고, Numerai Crypto 메타 모델을 사용하지 않는다. 보상은 Numerai의 재량에 따라, 사용자에게 공개되지 않을 수 있는 블랙박스 타킷을 기준으로 지급된다. 본질적으로 Numerai Crypto는 사용자가 자신의 CryptoSignals의 가치를 평가할 수 있도록 NMR 스테이킹을 통한 “진짜 시그널” 검증 수단을 제공하는 서비스다. 이러한 기대와 상이한 사용자는 스테이킹을 하지 말아야 한다.

자세한 내용은 [이용약관\(Terms of Service\)](#)을 참고하라.

FAQ

토큰 유니버스(token universe)란?

거래 가능한 잘 알려진 암호화폐 토큰 심볼의 집합이다. Numerai는 이 심볼들에 대한 예측만을 원한다. 자세한 내용은 [여기](#)를 참고하라.

Numerai Tournament, Numerai Signals, Numerai Crypto의 차이는?

Numerai Tournament와 Numerai Signals는 **주식시장** 예측 대회이고, Numerai Crypto는 **암호화폐 시장** 예측을 위한 대회다. Numerai Crypto는 사용자 고유의 피처를 사용해야 한다는 점에서 Signals와 가장 유사하다. Numerai는 모델 학습 및 검증 데이터 평가를 위한 [히스토리컬 타킷](#) 파일을 제공하지만, 피처는 사용자가 직접 수집·가공해야 한다.

내 코드/데이터가 Numerai에 노출되나?

아니다. Numerai는 사용자의 데이터·예측을 생성하는 코드를 열람하지 않는다. Numerai는 오직 **예측 결과**만 수신하며, 코드를 역추적하는 것은 사실상 불가능하고 그럴 의사도 없다.

참여를 위해 NMR을 반드시 스테이킹해야 하나?

아니다. 스테이킹 없이도 예측 파일을 제출하고 성과를 확인할 수 있다.

그렇다면 직접 거래하면 되지 않나?

물론 가능하며, 이를 막을 수 없다. 다만 Numerai는 사용자의 시그널이 이미 널리 알려져 모두가 사용하는 것인지 판단할 수 있으며, 모두가 같은 시그널로 거래한다면 그 가치는 낮다. 자세한 내용은 [이 Medium 글](#)을 참고하라.

참여하려면 코딩/금융 지식이 필요한가?

시작을 위한 예시 모델과 탐색용 노트북이 제공되지만, 코딩과 금융에 익숙하지 않은 사람은 Numerai Crypto보다는 [Numerai Tournament](#)부터 시작하는 것이 권장된다. Numerai Crypto에서 성공하려면 높은 수준의 코딩 및 시장 지식이 필요하다.

Numerai는 암호화폐를 거래하나?

아니다. Numerai도, Numerai의 헤지펀드도 암호화폐를 거래하지 않는다.

Numerai는 Numerai Crypto 메타 모델을 사용하나?

아니다. Numerai의 다른 제품, 자산, 서비스 어느 것도 Numerai Crypto 메타 모델을 소비·사용·혼합하지 않으며, 그 영향도 받지 않는다.

Numerai Crypto의 예측이나 데이터가 Numerai의 헤지펀드에 사용되나?

아니다. Numerai의 헤지펀드는 Numerai Crypto와 완전히 분리되어 있으며, Numerai Crypto가 헤지펀드에 어떤 영향도 주지 않는다.

지원 (Support)

질문, 지원, 피드백은 [Discord](#)에서 확인하라!

데이터 (Data)

데이터 구조 (Data Structure)

Numerai Crypto 데이터셋은 토큰 유니버스와 그들의 시간에 따른 수익률을 설명하는 테이블 형식의 데이터다.

각 행(row)은 특정 시점의 특정 토큰을 나타내며, `symbol`과 `date`로 식별된다.

`date`는 피쳐 생성을 위해 “마감 시점”(해당 일의 23:59 UTC)의 데이터가 사용된 날짜를 의미한다 — 라운드/라이브 데이터 맥락에서는 라운드가 열리기 전날을 뜻한다.

`target`은 해당 `date` 대비 미래 수익률(예: 30일 후)을 측정한다.

타겟 (Target)

Numerai Crypto에는 단일 타겟이 있으며, 이는 30일 후 토큰의 수익률이다. 각 라운드마다 유니버스 내 모든 토큰에 대해 30일 수익률을 순위화(ranked) 하고 가우시안화(gaussianized) 한 뒤 5개 구간(bin)으로 버킷팅한다.

데이터 API (Data API)

Numerai Crypto 데이터셋에 접근하는 가장 좋은 방법은 데이터 API를 사용하는 것이다:

```

from numerapi import NumerAPI
napi = NumerAPI()

[f for f in napi.list_datasets() if f.startswith("crypto/v1.0")]

[
    'crypto/v1.0/live_universe.parquet',
    'crypto/v1.0/train_targets.parquet',
]

# Download the training data
napi.download_dataset("crypto/v1.0/train_targets.parquet")

```

고품질 시그널을 생성하려면, 고유하고 구별 가능한 암호화폐 시장 데이터를 직접 확보해야 한다. 시작을 위해 사용할 수 있는 다른 데이터 제공자로는 [Messari](#)와 [CoinMarketCap](#)가 있다.

암호화폐 시장 데이터가 없지만 Numerai에 관심이 있다면, 우리의 데이터를 사용해 주식시장을 예측하는 [Numerai Tournament](#)를 대신 시도해 보라.

Modeling

데이터 모델링을 어디서 시작해야 할지 모르겠다면, Numerai Tournament의 [기본 모델링](#)을 먼저 학습 하길 권한다. 모델링에 자신이 생기면 그 기술은 Numerai Crypto에도 그대로 적용된다. 다음은 트리 기반 모델의 기본 예시다:

```

import lightgbm as lgb
import pandas as pd
import random
from numerapi import NumerAPI
from typing import List

napi = NumerAPI()

def generate_training_features(df: pd.DataFrame) -> List[str]:
    # TODO: Get your data and create features
    df['fake_feature_1'] = df.groupby(["symbol", "date"])
    ['symbol'].transform(lambda x: random.uniform(0, 1))
    return ['fake_feature_1']

# Historical targets file contains ["symbol", "date", "target"] columns
napi.download_dataset("crypto/v1.0/train_targets.parquet")

```

```
train_df = pd.read_parquet("crypto/v1.0/train_targets.parquet")

# Add training features for each (symbol, date)
feature_cols = generate_training_features(train_df)

model = lgb.LGBMRegressor(
    n_estimators=2000,
    learning_rate=0.01,
    max_depth=5,
    num_leaves=2 ** 5,
    colsample_bytree=0.1
)

model.fit(
    train_df[feature_cols],
    train_df["target"]
)
```

Submissions

Numerai Crypto 제출 방식은 Numerai Signals(참고: [here](#))와 매우 유사하지만, 주식 티커 대신 토큰 심볼을 사용한다. 제출물은 **수익률**을 예측하는 각 심볼과 하나의 수치 값을 가진 목록이어야 한다:

symbol	signal
BTC	0.529851
ETH	0.376866
SOL	0.272388
NMR	0.929104

제출에 포함되는 심볼 목록은 **Numerai Crypto 토큰 유니버스**로 정의된다. 또한 수치 값은 0과 1 사이여야 한다.

다음은 Python으로 실시간 예측을 생성하고 업로드하는 방법의 예시다:

```
def generate_features(df: pd.DataFrame):
    # TODO: Get your data and create features for live universe
    df['fake_feature_1'] = df['symbol'].transform(lambda x:
random.uniform(0, 1))

# Use API keys to authenticate
napi = NumerAPI("[your api public id]", "[your api secret key]")

# Download latest live universe
napi.download_dataset("crypto/v1.0/live_universe.parquet")
live = pd.read_parquet("crypto/v1.0/live_universe.parquet")

# Generate features for the live universe
generate_features(live)
```

```
# Get live predictions
live["signal"] = model.predict(live[feature_cols])

# Predictions must be between 0 and 1
live["signal"] = live["signal"].rank(pct=True)

# Format and save submission
live[['symbol', 'signal']].to_parquet("submission.parquet")

# Get model ids and submit models
models = napi.get_models(tournament=12)
for model_name, model_id in models.items():
    print(f'submitting {model_name}...')
    napi.upload_predictions("submission.parquet", model_id=model_id,
tournament=12)

print('done!')
```

제출에 대해 더 알아보려면 [here](#)를 참고해라.

Scoring

채점은 Numerai Tournament(참고: [here](#))와 매우 유사하다.

Numerai Crypto는 단지 토큰 수익률을 예측하는 것에 그치지 않고, 다른 모델에 아직 존재하지 않는 **독창적 시그널**을 찾는 데 목적이 있다.

자세한 내용은 [Scoring](#) 섹션을 참고해라.

Staking

Numerai Tournament와 마찬가지로, 성과에 따라 수익 또는 소각이 이루어지도록 모델에 선택적으로 stake [NMR](#)을 할 수 있다. 자세한 내용은 [the Staking docs](#)를 읽어라.

시그널을 스테이킹할 기회는 Numerai가 제안하는 투자 계약, 증권, 어떤 금융자산의 수익률에 기반한 스왑, Numerai의 헤지펀드 지분, 혹은 Numerai 자체나 우리가 벌어들이는 수수료에 대한 참여를 의미하지 않는다. Numerai의 헤지펀드는 Numerai Crypto와 아무런 관련이 없으며, 암호화폐를 거래하지도 않고, Numerai Crypto 메타 모델을 사용하지도 않는다. 보상은 당사의 재량에 따라, 사용자에게 공개되지 않을 수 있는 블랙박스 타깃을 기반으로 지급될 수 있다. 근본적으로, Numerai Crypto는 사용자가 자신의 CryptoSignals의 가치를 평가할 수 있도록 NMR 스테이킹을 “진짜” 시그널 검증 수단으로 활용하는 서비스를 제공한다. 이와 다른 기대를 가진 사용자는 스테이킹을 하지 않아야 한다.

자세한 내용은 우리의 [Terms of Service](#) 를 읽어라.

FAQ

토큰 유니버스란 무엇인가?

거래 가능한 잘 알려진 암호화폐 토큰 심볼의 집합이다. Numerai는 이러한 심볼에 대한 예측만을 원한다. 자세한 내용은 [here](#)를 참고해라.

Numerai Tournament, Numerai Signals, Numerai Crypto의 차이는 무엇인가?

Numerai Tournament와 Numerai Signals는 사용자가 주식시장을 예측하는 대회다. Numerai Crypto는 암호화폐 시장 예측을 위한 것이다. Numerai Crypto는 사용자가 자체 피처를 사용해야 한다는 점에서 Signals와 가장 유사하다. Numerai는 모델 학습 및 검증 데이터 평가를 위한 [historical targets](#) 파일을 제공하지만, 피처는 사용자가 직접 수집하고 처리해야 한다.

제 코드/데이터는 Numerai로부터 보호되나?

그렇다. Numerai는 사용자의 데이터나 예측을 생성하는 코드를 열람하지 않는다. Numerai는 오직 예측 결과만을 받는다. 사용자의 코드를 역설계하는 것은 사실상 불가능하며, Numerai는 그럴 의사도 없다.

참여하려면 NMR을 반드시 스테이킹해야 하나?

아니다. 스테이킹 없이도 예측 파일을 제출하고 성과를 받을 수 있다.

차라리 직접 거래하면 안 되나?

가능하고, 우리가 이를 막을 수는 없다. 다만 Numerai는 사용자의 시그널이 이미 알려져 모두가 사용 중인 지 여부를 파악할 수 있으며, 모두가 거래하는 시그널로 거래하는 것은 그리 가치가 높지 않다. [이 Medium 글](#)을 참고해라.

참여하려면 코딩/금융 지식이 필요한가?

시작을 위한 예시 모델과 탐색용 노트북이 제공되지만, 코딩과 금융에 익숙하지 않다면 Numerai Crypto 보다는 [Numerai Tournament](#)를 권한다. Numerai Crypto에서 성공하려면 높은 수준의 코딩 및 시장 지식이 필요하다.

Numerai는 암호화폐를 거래하나?

아니다. Numerai도, Numerai의 헤지펀드도 암호화폐를 거래하지 않는다.

Numerai는 Numerai Crypto 메타 모델을 사용하나?

아니다. Numerai의 다른 어떤 제품, 자산, 서비스도 Numerai Crypto 메타 모델을 소비하거나, 사용하거나, 혼합하거나, 그 영향하에 있지 않다.

Numerai Crypto의 예측이나 데이터가 Numerai의 헤지펀드에서 사용되나?

아니다. Numerai의 헤지펀드는 Numerai Crypto와 완전히 분리되어 있으며, Numerai Crypto가 헤지펀드에 어떤 영향도 미치지 않는다.

Support

질문, 지원, 피드백은 [Discord](#)에서 찾아봐라!

Data

Data Structure

Numerai Crypto 데이터셋은 토큰 유니버스와 그들의 시계열 수익률을 설명하는 테이블 형식의 데이터다.

상위 수준에서 각 행은 `symbol`과 `date`로 식별되는 특정 시점의 특정 토큰을 나타낸다. `date`는 피처 생성을 위해 “마켓 클로즈”(해당 일 23:59 UTC) 데이터가 사용된 날짜를 의미한다 — 라운드 및 라이브 데이터 맥락에서는 라운드가 열리기 전날이다. `target`은 해당 `date` 대비 미래 수익률(예: 30일 후)을 측정한 값이다.

Target

Numerai Crypto에는 하나의 타겟이 있으며, 이는 30일 후 토큰의 수익률이다. 각 라운드마다 유니버스 내 각 토큰의 30일 수익률을 순위화하고, 가우시안화한 다음, 다섯 개의 구간으로 버킷팅한다.

Data API

Numerai Crypto 데이터셋에 접근하는 가장 좋은 방법은 데이터 API를 사용하는 것이다:

```
from numerapi import NumerAPI
napi = NumerAPI()

[f for f in napi.list_datasets() if f.startswith("crypto/v1.0")]

[
    'crypto/v1.0/live_universe.parquet',
    'crypto/v1.0/train_targets.parquet',
]

# Download the training data
napi.download_dataset("crypto/v1.0/train_targets.parquet")
```

- `train_targets.parquet` 에는 과거 심볼과 타겟이 포함된다
- `live_universe.parquet` 에는 현재 라운드의 타겟이 없는 최신 토큰 유니버스가 포함된다

Community Data

Numerai Council of Elders는 [YIEDL](#)과 협력하여 YIEDL 크립토 데이터셋을 Numerai Crypto에 도입했다. 이 포괄적 컬렉션은 10년이 넘는 암호화폐 데이터를 포함하며, 가격·거래량·모멘텀(PVM), 감성 분석, 온체인 지표와 같은 구성요소를 제공한다.

- [Official Announcement](#) 읽어라
- [Example Notebook](#) 실행해라

Submissions

CryptoSignals을 생성하는 데 사용되는 근본 데이터는 매우 다를 수 있지만(감사 재무제표 vs 뉴스 기사 vs 주차장 이미지), CryptoSignals 자체는 모두 동일한 기본 형식을 따른다 — 각 토큰 심볼과 그에 대응하는 하나의 수치 값 목록:

symbol	signal
BTC	0.529851
ETH	0.376866
SOL	0.272388
NMR	0.929104

제출물에 포함되는 토큰 심볼 목록은 **Numerai Crypto 토큰 유니버스**로 정의된다. 이는 전 세계 상위 약 300개 토큰을 포괄하며 매일 업데이트된다. 일반적으로 거래량이 낮은 일부 토큰만 유입되거나 제외되지만, 이는 시장 변동성에 좌우되며(잘 알려졌듯이) 크립토는 매우 변동성이 클 수 있다.

최신 유니버스는 `live.parquet` 파일을 다운로드하여 확인할 수 있다.

토큰 유니버스는 모든 토큰을 검토한 다음 다음과 같이 정의된다:

1. 항상 BTC, ETH, NMR을 유니버스에 포함
2. 스테이블코인, 래핑, 리퀴드 스테이킹(예: stETH, rpETH) 태그가 있는 심볼 제거
3. 데이터 포인트가 30개 미만이고 생성된 지 2년 미만인 심볼 제거
4. 최신 시가총액이 0인 심볼 제거
5. 24시간 거래량이 \$500,000 미만인 심볼 제거
6. 최신 시가총액이 더 낮은 중복 심볼 제거
7. `min_latest_volume` 미만의 거래량을 가진 심볼 제거
8. 절대 평균 수익률이 0.00001 미만인 심볼 제거

9. 더 높은 시가총액 토큰과 상관계수(≥ 0.95)가 매우 높은 심볼 제거
10. 시가총액 기준 상위 500개 토큰 반환

시그널을 제출할 때는 최소 두 개의 열이 필요하다:

- `symbol` 열 — 값은 라이브 유니버스 내의 유효한 심볼이어야 한다.
- `signal` 열 — 값은 0과 1 사이(양끝 제외)여야 한다.

추가로, `live` 제출이 유효하려면 다음 조건을 만족해야 한다:

- 유니버스에서 적어도 100개 이상의 심볼이 0~1 범위의 유효 값을 가져야 한다.
- 하나의 심볼은 한 번만 나타날 수 있다.

제출 창이 어떻게, 언제 열리고 닫히는지 기본 사항은 [Numerai Tournament Submissions docs](#)를 참고해라.

API를 사용해 제출 워크플로를 자동화할 수 있다. 제출 API는 Parquet과 CSV 파일 형식을 모두 지원한다. 다음은 NumerAPI로 제출을 업로드하는 예시다:

```
from numerapi import CryptoAPI

# Authenticate
napi = CryptoAPI("[your api public id]", "[your api secret key]")
napi.upload_predictions("[path to your submission]", model_id="your-model-id")
```

- 다음 중 아무 것이나 사용할 수 있다
 - [Numerapi](#) (공식 Python 클라이언트)
 - 기타 언어용 원시 [GraphQL API](#)
- 우리의 오픈소스 클라우드 자동화 도구 [Numerai Compute](#)

Scoring

Numerai Crypto에는 두 가지 주요 점수가 있다

- [Meta Model Contribution](#) (MMC): Crypto 메타 모델에 대한 예측 기여도
- [Correlation](#) (CORR): 예측과 타겟의 상관관계

자세한 설명의 전체 목록은 [definitions docs](#)를 참고해라.

Definitions

아래에 참조된 기본 함수들을 이해하려면 [main definitions docs](#)를 참고해라.

Targets

- target
 - 타임라인: 20D2L
 - bins=5, uniformity=10%, 40%, 50%

Meta Models

- Numerai Crypto 제출물은 먼저 다음과 같이 정리된다:
 - 토큰 유니버스와 조인
 - nan 및 중복 심볼 제거
 - 각 제출물에 tie-kept-rank 적용
 - 각 제출물의 nan을 0.5로 채움
- 최소 1 NMR 스테이크 조건의 Numerai Crypto 단순 가중 메타 모델(CNWMMmin1)
 - 최소 1 NMR 스테이크가 있는 정리된 Crypto 제출물의 평균

Scores

- 채점에 사용되기 전에 제출물은 다음과 같이 정리된다:
 - 유효하지 않은 심볼 제거
 - 제출물에 tie-kept rank 적용
 - nan을 0.5로 채움
- MMC - Meta Model Contribution
 - 제출물, CNWMMmin1, target의 상관 기여도
 - 타임라인: 20D2L (+ 2일 데이터 지연)
- CORR - Correlation
 - 제출물과 target의 numerai corr
 - 20 채점일 및 2일 수익률 지연(+ 2일 데이터 지연)

Staking

시그널에 [churn and turnover](#)가 충분히 낮다면, 점수에 따라 수익 또는 소각이 이루어지도록 모델에 선택적으로 [stake NMR](#)을 할 수 있다. 스테이킹은 사용자의 NMR을 잠그고, 잠긴 NMR에 대해 Numerai가 보상을 추가하거나 소각할 수 있도록 허용한다는 의미다. 스테이킹과 보상에 대한 일반 규칙은 [main staking documentation](#)를 읽어라.

시그널을 스테이킹할 기회는 Numerai가 제안하는 투자 계약, 증권, 어떤 금융자산의 수익률에 기반한 스왑, Numerai의 헤지펀드 지분, 혹은 Numerai 자체나 우리가 벌어들이는 수수료에 대한 참여를 의미하지 않는다. 보상은 사용자에게 공개되지 않을 블랙박스 타킷을 기반으로, 당사의 재량에 따라 지급된다. 근본적으로, Numerai Signals는 NMR 스테이킹을 “진짜” 시그널 검증 수단으로 활용하여 사용자가 시그널의 가치를 평가할 수 있도록 하는 서비스다. 그 대가로 Numerai는 스테이킹된 시그널과 관련 데이터를 Numerai 헤지펀드에서 활용한다. 이러한 기대와 다른 사용자는 시그널을 스테이킹하지 않아야 한다.

자세한 내용은 우리의 [Terms of Service](#) 를 읽어라.

Payouts

보상은 스테이크 값과 점수의 함수다. 보상이 일반적으로 어떻게 작동하는지 이해하려면 [main staking documentation](#)를 검토해라. Signals 보상 함수는 구체적으로 다음과 같다:

$$\text{payout} = \text{stake} * \text{clip}(\text{payout_factor} * \text{mmc}), -0.05, 0.05)$$

`payout_factor`는 총 스테이킹된 NMR이 `stake_cap_threshold`를 초과하여 증가함에 따라 로그 형태로 감소한다. `stake_cap_threshold`의 값은 [here](#)에서 확인해라.

Connect

Index

- Business Inquiries - contact@numer.ai
- Technical Support - support@numer.ai
- Security Reports - security@numer.ai
- Fund Investment - investing@numer.ai
- Official Discord

- Channel - <https://discord.com/numerai>
- Admin - the_arkivist (ark)

Security Notice

Numerai는 위의 채널을 통해서만 소통한다. Numerai는 전화나 SMS, 또는 공식 Discord 채널 외의 메시징 플랫폼을 통해 먼저 연락하지 않는다. 우리는 비공식 채널을 통한 서비스 제안이나 아웃리치를 적극적으로 하지 않는다. Numerai 소속이라고 주장하는 개인으로부터 예기치 않게 연락을 받으면, security@numer.ai로 신고해라.