```
# This Python 3 environment comes with many helpful analytics
libraries installed
# It is defined by the kaggle/python Docker image:
https://github.com/kaggle/docker-python # For example,
here's several helpful packages to load

import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)

# Input data files are available in the read-only "../input/"
directory
# For example, running this (by clicking run or pressing Shift+Enter)
will list all files under the input directory

import os for dirname, _, filenames in
os.walk('/kaggle/input'):     for filename in
filenames:         print(os.path.join(dirname,
filename))

# You can write up to 20GB to the current directory (/kaggle/working/)
that gets preserved as output when you create a version using "Save &
Run All"
# You can also write temporary files to /kaggle/temp/, but they won't
be saved outside of the current session

/kaggle/input/hull-tactical-market-prediction/train.csv
/kaggle/input/hull-tactical-market-prediction/test.csv
/kaggle/input/hull-tactical-market-prediction/kaggle_evaluation/defaul
t_inference_server.py
/kaggle/input/hull-tactical-market-prediction/kaggle_evaluation/defaul
t_gateway.py
/kaggle/input/hull-tactical-market-prediction/kaggle_evaluation/__init
__.py
/kaggle/input/hull-tactical-market-prediction/kaggle_evaluation/core/
templates.py
/kaggle/input/hull-tactical-market-prediction/kaggle_evaluation/core/
base_gateway.py
/kaggle/input/hull-tactical-market-prediction/kaggle_evaluation/core/
relay.py
/kaggle/input/hull-tactical-market-prediction/kaggle_evaluation/core/
kaggle_evaluation.proto
/kaggle/input/hull-tactical-market-prediction/kaggle_evaluation/core/
__init__.py
/kaggle/input/hull-tactical-market-prediction/kaggle_evaluation/core/
generated/kaggle_evaluation_pb2.py
/kaggle/input/hull-tactical-market-prediction/kaggle_evaluation/core/
generated/kaggle_evaluation_pb2_grpc.py
/kaggle/input/hull-tactical-market-prediction/kaggle_evaluation/core/
generated/__init__.py
```

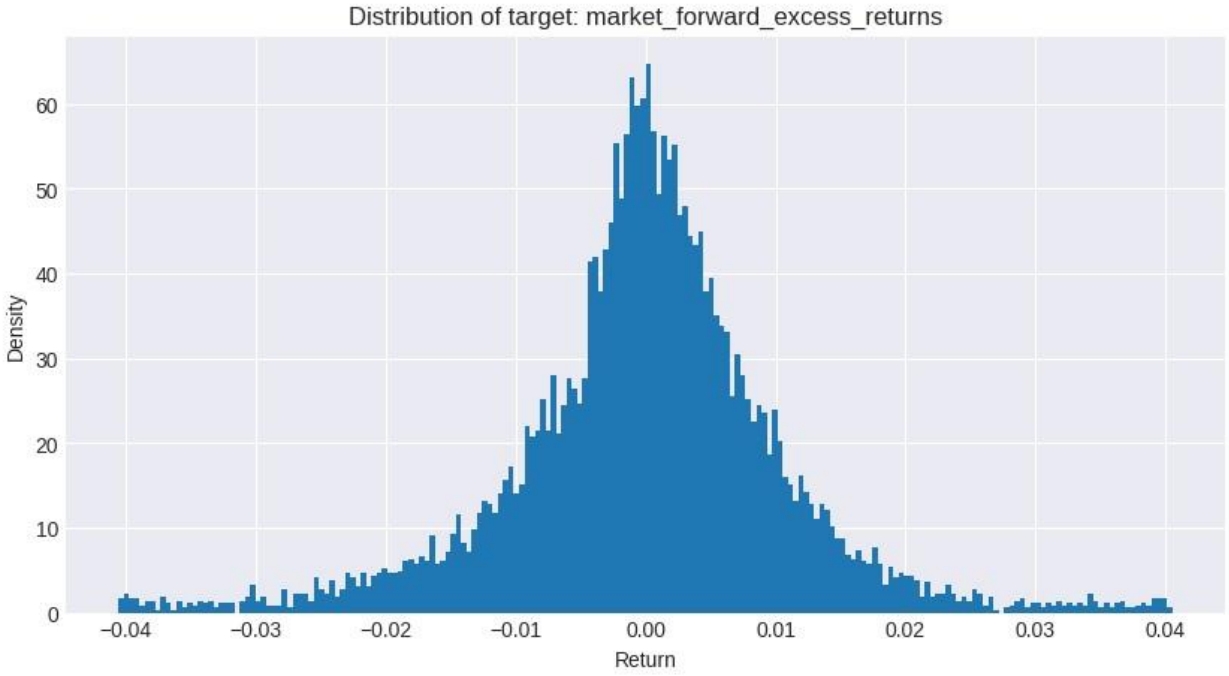
```
!pip install pandas numpy matplotlib seaborn plotly scikit-learn scipy
statsmodels tqdm

# -------------------------
# 0) Imports & config
# -------------------------
import os import warnings
warnings.filterwarnings("ignore")

import numpy as np
import pandas as pd
from scipy import stats
from tqdm import tqdm

import matplotlib.pyplot as plt
import seaborn as sns import
plotly.express as px import
plotly.graph_objects as go

from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA

# plotting defaults
plt.style.use('seaborn-darkgrid')
plt.rcParams['figure.figsize'] = (12,6)

# -------------------------
# 1) Load data
# -------------------------
DATA_DIR = "/kaggle/input/hull-tactical-market-prediction"
TRAIN_PATH = os.path.join(DATA_DIR, "train.csv")
TEST_PATH = os.path.join(DATA_DIR, "test.csv")

train = pd.read_csv(TRAIN_PATH)
test = pd.read_csv(TEST_PATH)

print('Train shape:', train.shape)
print('Test shape:', test.shape)
print('Sample train columns:', train.columns[:40].tolist())

# -------------------------
# 2) Preprocessing (EDA-friendly)
# -------------------------
TARGET = 'market_forward_excess_returns' if
'market_forward_excess_returns' in train.columns else
'forward_returns'
print('Target chosen:', TARGET)

train_sorted = train.sort_values('date_id').reset_index(drop=True)
```

```
# Fill strategy: forward-fill then backward-fill, then median
feature_cols = [c for c in train.columns if c not in
['date_id','forward_returns','risk_free_rate','market_forward_excess_r
eturns']]
train_ffill = train_sorted.copy()
train_ffill[feature_cols] = train_ffill[feature_cols].ffill().bfill()
train_ffill.fillna(train_ffill.median(numeric_only=True),
inplace=True)

# Winsorize extreme values
num_cols =
train_ffill[feature_cols].select_dtypes(include=[np.number]).columns.t
olist() for c in num_cols:     low = train_ffill[c].quantile(0.005)
high = train_ffill[c].quantile(0.995)
    train_ffill[c] = train_ffill[c].clip(lower=low, upper=high)

# -------------------------
# 3) Exploratory Data Analysis
# -------------------------

# 3.1 Target distribution
plt.figure(figsize=(10,5))
plt.hist(train_ffill[TARGET].dropna(), bins=200, density=True)
plt.title(f'Distribution of target: {TARGET}')
plt.xlabel('Return'); plt.ylabel('Density') plt.show()

# 3.2 Rolling mean and volatility of target
plt.figure(figsize=(12,6))
ax = train_ffill[TARGET].rolling(252).mean().plot(label='252d mean')
train_ffill[TARGET].rolling(252).std().plot(label='252d vol', ax=ax)
ax.set_title('252-day rolling mean & volatility of target')
ax.legend(); plt.show()

# 3.3 Missingness heatmap for top features
present_pct = train_ffill.notna().mean()
top_feats =
present_pct.sort_values(ascending=False).index[:50].tolist()
sns.heatmap(train_sorted[top_feats].isnull().T, cbar=False)
plt.title('Missingness (top features)') plt.xlabel('Row
index'); plt.ylabel('Features') plt.show()

# 3.4 Correlation heatmap (sampled subset to reduce load)
corr_feats = train_ffill[num_cols].sample(frac=0.15, random_state=0)
corr = corr_feats.corr() plt.figure(figsize=(12,10))
```

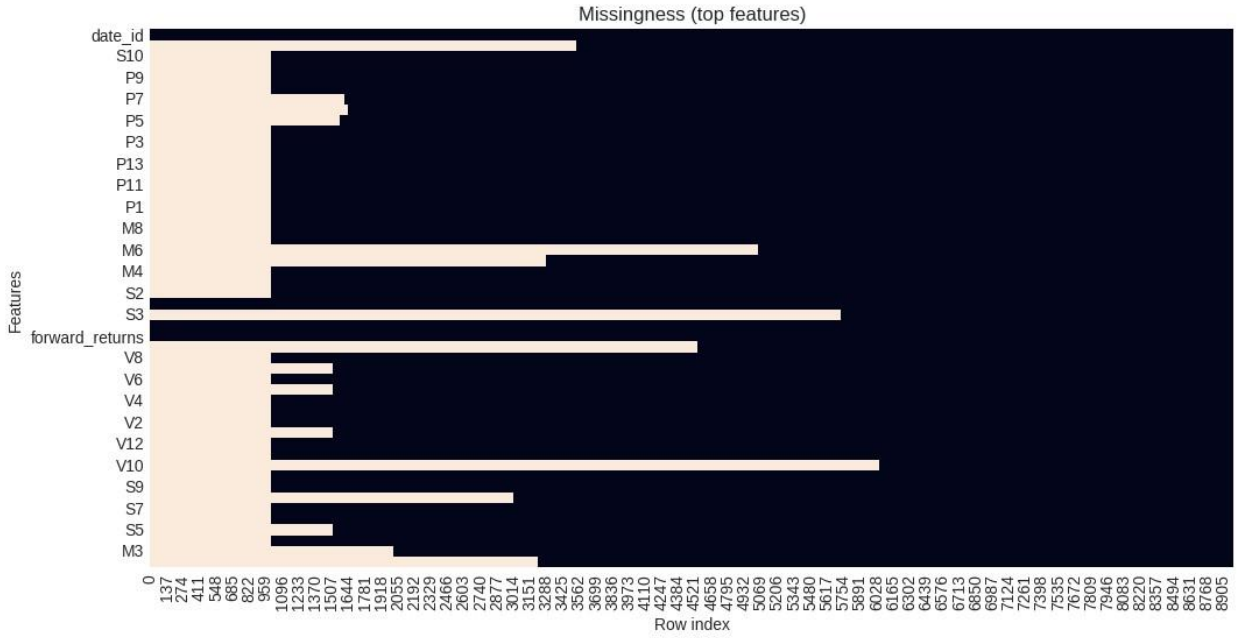
```
sns.heatmap(corr, cmap='coolwarm', center=0)
plt.title('Feature correlation (sampled subset)')
plt.show()

# 3.5 PCA overview scaler
= StandardScaler()
X_scaled = scaler.fit_transform(train_ffill[num_cols].fillna(0))
pc = PCA(n_components=6) pc_comp = pc.fit_transform(X_scaled)

print('\nExplained variance ratio (first 6 PCs):',
pc.explained_variance_ratio_)
plt.bar(range(1,7), pc.explained_variance_ratio_)
plt.xlabel('PC'); plt.ylabel('Explained variance')
plt.title('PCA explained variance (first 6 PCs)')
plt.show()

# 3.6 Interactive Plotly example: cumulative returns
train_ffill['cum_return'] = (1 +
train_ffill[TARGET].fillna(0)).cumprod()
fig = px.line(train_ffill, x='date_id', y='cum_return',
title='Cumulative returns over time') fig.show()

print("\n EDA complete — dataset profiled, missingness visualized,
correlations explored, PCA variance checked, and target dynamics
plotted.")

Requirement already satisfied: pandas in
/usr/local/lib/python3.11/dist-packages (2.2.3)
Requirement already satisfied: numpy in
/usr/local/lib/python3.11/dist-packages (1.26.4)
Requirement already satisfied: matplotlib in
/usr/local/lib/python3.11/dist-packages (3.7.2)
Requirement already satisfied: seaborn in
/usr/local/lib/python3.11/dist-packages (0.12.2)
Requirement already satisfied: plotly in
/usr/local/lib/python3.11/dist-packages (5.24.1)
Requirement already satisfied: scikit-learn in
/usr/local/lib/python3.11/dist-packages (1.2.2)
Requirement already satisfied: scipy in
/usr/local/lib/python3.11/dist-packages (1.15.3)
Requirement already satisfied: statsmodels in
/usr/local/lib/python3.11/dist-packages (0.14.4)
Requirement already satisfied: tqdm in /usr/local/lib/python3.11/dist-
packages (4.67.1)
Requirement already satisfied: python-dateutil>=2.8.2 in
/usr/local/lib/python3.11/dist-packages (from pandas) (2.9.0.post0)
Requirement already satisfied: pytz>=2020.1 in
/usr/local/lib/python3.11/dist-packages (from pandas) (2025.2)
Requirement already satisfied: tzdata>=2022.7 in
```

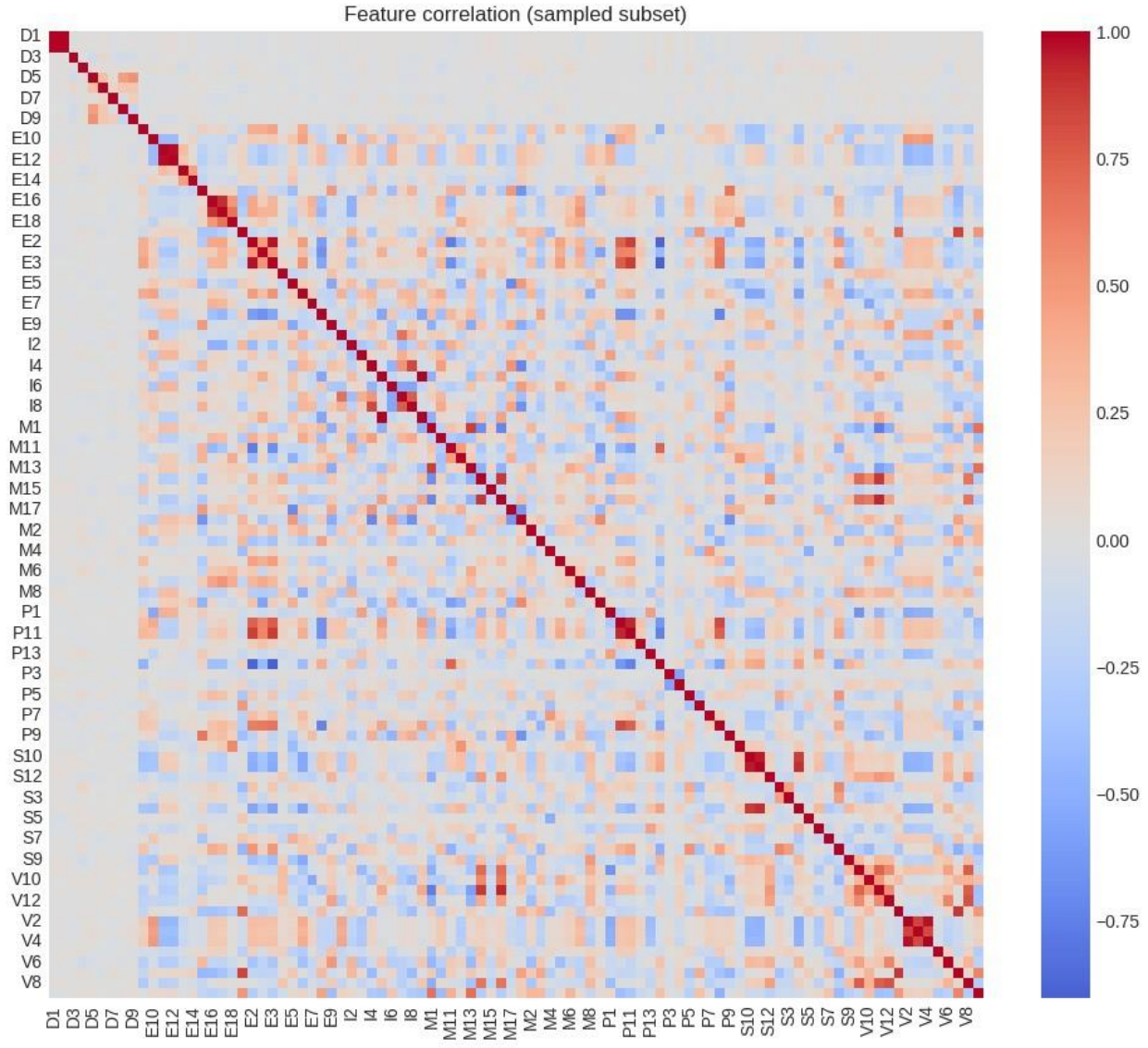
```
/usr/local/lib/python3.11/dist-packages (from pandas) (2025.2)
Requirement already satisfied: mkl_fft in
/usr/local/lib/python3.11/dist-packages (from numpy) (1.3.8)
Requirement already satisfied: mkl_random in
/usr/local/lib/python3.11/dist-packages (from numpy) (1.2.4)
Requirement already satisfied: mkl_umath in
/usr/local/lib/python3.11/dist-packages (from numpy) (0.1.1)
Requirement already satisfied: mkl in /usr/local/lib/python3.11/dist-
packages (from numpy) (2025.2.0)
Requirement already satisfied: tbb4py in
/usr/local/lib/python3.11/dist-packages (from numpy) (2022.2.0)
Requirement already satisfied: mkl-service in
/usr/local/lib/python3.11/dist-packages (from numpy) (2.4.1)
Requirement already satisfied: contourpy>=1.0.1 in
/usr/local/lib/python3.11/dist-packages (from matplotlib) (1.3.2)
Requirement already satisfied: cycler>=0.10 in
/usr/local/lib/python3.11/dist-packages (from matplotlib) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in
/usr/local/lib/python3.11/dist-packages (from matplotlib) (4.58.4)
Requirement already satisfied: kiwisolver>=1.0.1 in
/usr/local/lib/python3.11/dist-packages (from matplotlib) (1.4.8)
Requirement already satisfied: packaging>=20.0 in
/usr/local/lib/python3.11/dist-packages (from matplotlib) (25.0)
Requirement already satisfied: pillow>=6.2.0 in
/usr/local/lib/python3.11/dist-packages (from matplotlib) (11.2.1)
Requirement already satisfied: pyparsing<3.1,>=2.3.1 in
/usr/local/lib/python3.11/dist-packages (from matplotlib) (3.0.9)
Requirement already satisfied: tenacity>=6.2.0 in
/usr/local/lib/python3.11/dist-packages (from plotly) (8.5.0)
Requirement already satisfied: joblib>=1.1.1 in
/usr/local/lib/python3.11/dist-packages (from scikit-learn) (1.5.1)
Requirement already satisfied: threadpoolctl>=2.0.0 in
/usr/local/lib/python3.11/dist-packages (from scikit-learn) (3.6.0)
Requirement already satisfied: patsy>=0.5.6 in
/usr/local/lib/python3.11/dist-packages (from statsmodels) (1.0.1)
Requirement already satisfied: six>=1.5 in
/usr/local/lib/python3.11/dist-packages (from python-dateutil>=2.8.2-
>pandas) (1.17.0)
Requirement already satisfied: intel-openmp<2026,>=2024 in
/usr/local/lib/python3.11/dist-packages (from mkl->numpy) (2024.2.0)
Requirement already satisfied: tbb==2022.* in
/usr/local/lib/python3.11/dist-packages (from mkl->numpy) (2022.2.0)
Requirement already satisfied: tcmlib==1.* in
/usr/local/lib/python3.11/dist-packages (from tbb==2022.*->mkl->numpy)
(1.4.0)
Requirement already satisfied: intel-cmplr-lib-rt in
/usr/local/lib/python3.11/dist-packages (from mkl_umath->numpy)
(2024.2.0)
```

```
Requirement already satisfied: intel-cmplr-lib-ur==2024.2.0 in
/usr/local/lib/python3.11/dist-packages (from intel-
openmp<2026,>=2024->mkl->numpy) (2024.2.0)
Train shape: (8990, 98)
Test shape: (10, 99)
Sample train columns: ['date_id', 'D1', 'D2', 'D3', 'D4', 'D5', 'D6',
'D7', 'D8', 'D9', 'E1', 'E10', 'E11', 'E12', 'E13', 'E14', 'E15',
'E16', 'E17', 'E18', 'E19', 'E2', 'E20', 'E3', 'E4', 'E5', 'E6', 'E7',
'E8', 'E9', 'I1', 'I2', 'I3', 'I4', 'I5', 'I6', 'I7', 'I8', 'I9',
'M1']
Target chosen: market_forward_excess_returns
```

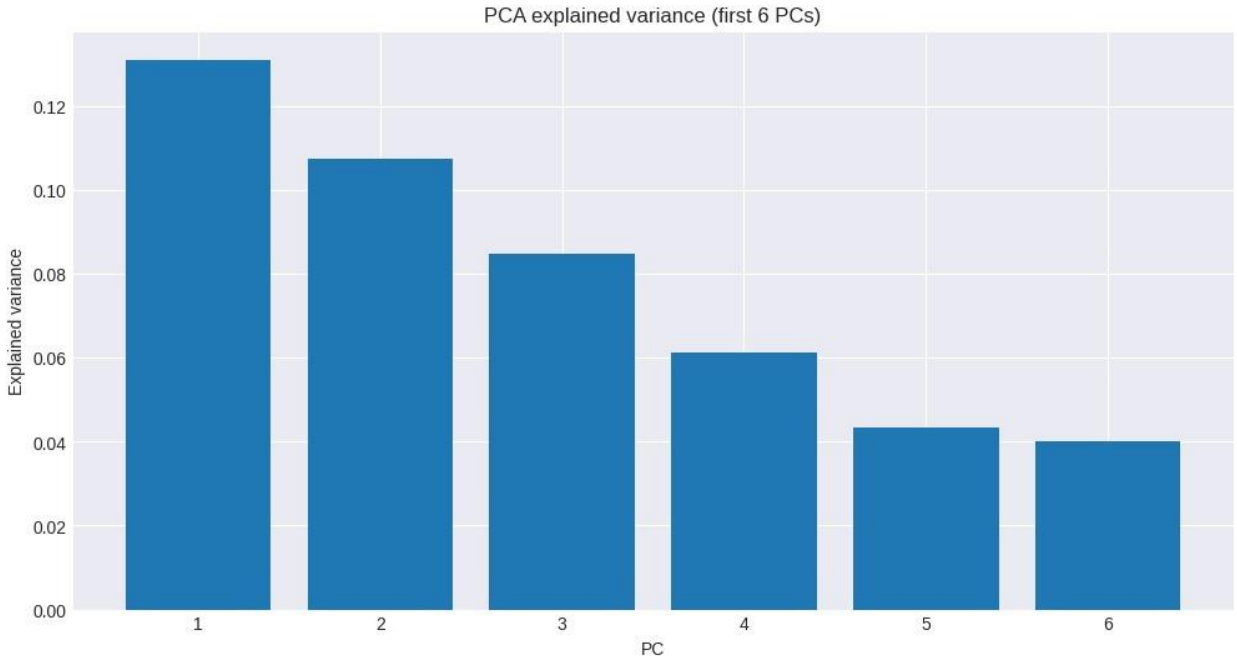



252-day rolling mean & volatility of target
252d mean
252d vol
0.020
0.015
0.010
0.005
0.000
0
2000
4000
6000
8000


Missingness (top features)
Features
date_id
S10
P9
P7
P5
P3
P13
P11
P1
M8
M6
M4
S2
S3
forward_returns
V8
V6
V4
V2
V12
V10
S9
S7
S5
M3
Row index




```
Explained variance ratio (first 6 PCs): [0.13108767 0.10745505
0.0847642  0.06110868 0.04329648 0.04013774]
```




```
 EDA complete — dataset profiled, missingness visualized, correlations
explored, PCA variance checked, and target dynamics plotted.

# -------------------------
# 4) Advanced EDA Extensions (20+)
# -------------------------
import numpy as np import
pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns from
scipy import stats
from statsmodels.graphics.tsaplots import plot_acf, plot_pacf
import statsmodels.api as sm

# 4.1 Autocorrelation of target try:
plot_acf(train_ffill[TARGET].dropna(), lags=60)
plt.title("Autocorrelation of target returns")
plt.show()

    plot_pacf(train_ffill[TARGET].dropna(), lags=60)
plt.title("Partial Autocorrelation of target returns")
plt.show() except Exception as e:     print("ACF/PACF
error:", e)

# 4.2 Rolling Sharpe ratio
try:
```

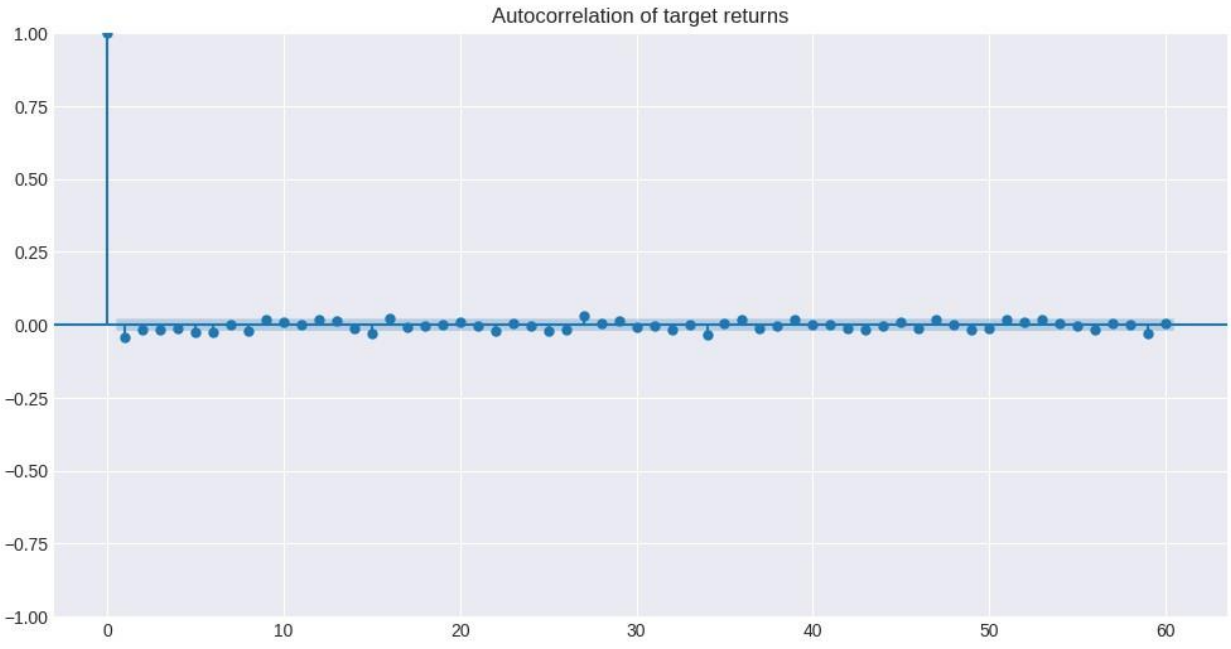
```
    rf = train_ffill.get("risk_free_rate", 0)
rolling_excess = train_ffill[TARGET] - rf
sharpe = rolling_excess.rolling(252).mean() /
(rolling_excess.rolling(252).std() + 1e-8)
sharpe.plot()
    plt.title("252-day rolling Sharpe ratio")
plt.show() except Exception as e:
print("Rolling Sharpe error:", e)

# 4.3 Volatility clustering (squared returns) try:
plt.plot(train_ffill['date_id'], train_ffill[TARGET]**2)
plt.title("Squared returns (volatility clustering)")
plt.show() except Exception as e:     print("Volatility
clustering error:", e)

# 4.4 QQ plot for target distribution try:
sm.qqplot(train_ffill[TARGET].dropna(), line='s')
plt.title("QQ Plot of target returns")     plt.show()
except Exception as e:     print("QQ plot error:", e)

# 4.5 Histogram with KDE overlay try:
sns.histplot(train_ffill[TARGET].dropna(), kde=True, bins=200)
plt.title("Distribution of target returns with KDE")
plt.show() except Exception as e:     print("Histogram KDE
error:", e)

# 4.6 Rolling correlation of target with V1 (volatility feature)
if "V1" in train_ffill.columns:     try:         rcorr =
train_ffill[TARGET].rolling(252).corr(train_ffill["V1"])
plt.plot(rcorr)
        plt.title("Rolling 252-day correlation of target with V1")
plt.show()     except Exception as e:         print("Rolling
correlation error:", e)

# 4.7 Lagged correlations
for lag in [1, 5, 21, 63]:
try:         corr =
train_ffill[TARGET].shift(lag).corr(train_ffill[TARGET])
```

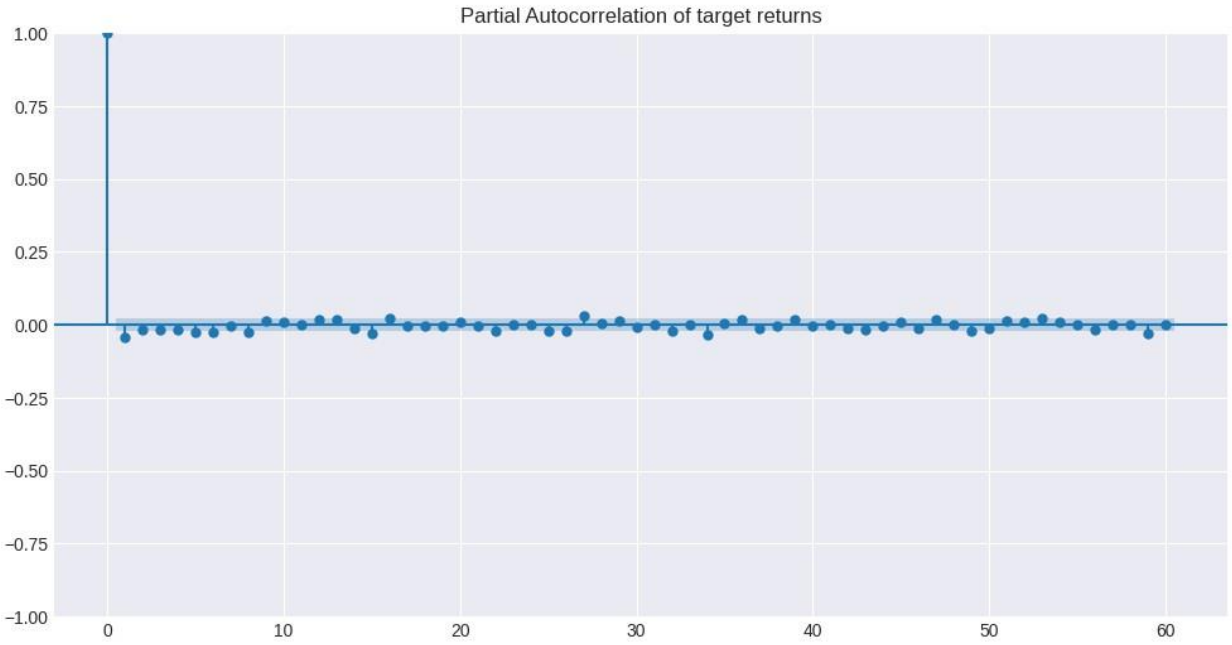
```
        print(f"Lag-{lag} correlation with target: {corr:.4f}")
except Exception as e:         print(f"Lag-{lag} correlation
error:", e)

# 4.8 Heatmap of feature importance (target correlation)
try:     target_corr =
train_ffill[num_cols].corrwith(train_ffill[TARGET]).sort_values(ascend
ing=False)
    sns.barplot(x=target_corr.head(20), y=target_corr.head(20).index)
plt.title("Top 20 features correlated with target")     plt.show()
except Exception as e:     print("Feature correlation heatmap
error:", e)

# 4.9 Interactive pairplot (subset) try:     subset_cols = [TARGET] +
num_cols[:5]     if train_ffill[subset_cols].dropna().shape[0] > 10:
sns.pairplot(train_ffill[subset_cols].dropna().sample(min(500,
train_ffill.shape[0]), random_state=0))         plt.show() except
Exception as e:     print("Pairplot error:", e)

# 4.10 Time series seasonal decomposition (monthly avg)
try:     if 'month' not in train_ffill.columns:
train_ffill['month'] = train_ffill['date_id'] % 12
monthly = train_ffill.groupby('month')[TARGET].mean()
monthly.plot(kind='bar')
    plt.title("Monthly seasonality of returns")
plt.show() except Exception as e:
print("Monthly seasonality error:", e)

# 4.11 Rolling skewness & kurtosis try:     from scipy.stats
import skew, kurtosis     roll_skew =
train_ffill[TARGET].rolling(252).apply(lambda x:
skew(x), raw=True)     roll_kurt =
train_ffill[TARGET].rolling(252).apply(lambda x: kurtosis(x),
raw=True)
    plt.plot(roll_skew, label="Skewness")
plt.plot(roll_kurt, label="Kurtosis")
    plt.legend(); plt.title("Rolling skewness & kurtosis of returns")
plt.show() except Exception as e:     print("Rolling skew/kurtosis
error:", e)
```

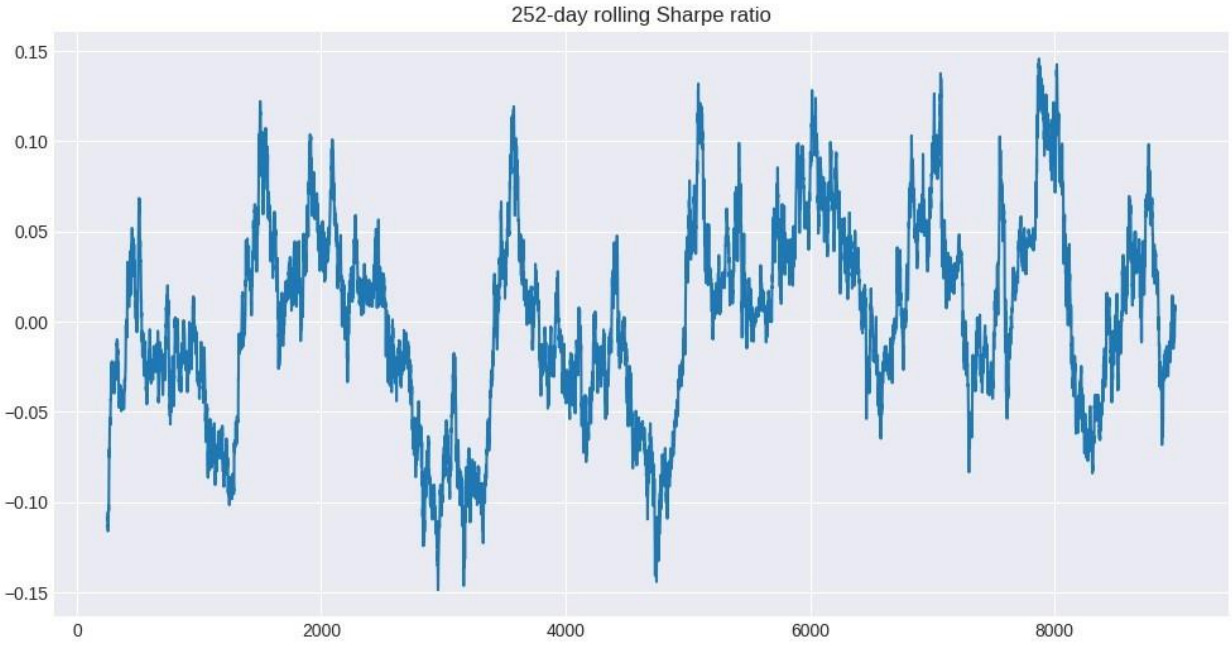
```
# 4.12 Correlation between sentiment (S*) and returns
try:
    sentiment_cols = [c for c in train_ffill.columns if
c.startswith("S")]
if sentiment_cols:
corr_sent =
train_ffill[sentiment_cols].corrwith(train_ffill[TARGET])
corr_sent.sort_values().plot(kind="barh")
        plt.title("Correlation of sentiment features with returns")
plt.show() except Exception as e:     print("Sentiment correlation
error:", e)

# 4.13 Cumulative distribution function try:
sns.ecdfplot(train_ffill[TARGET].dropna())
plt.title("Empirical CDF of returns")
plt.show() except Exception as e:
print("ECDF error:", e)

# 4.14 Drawdowns try:     cum = (1 +
train_ffill[TARGET].fillna(0)).cumprod()
rolling_max = cum.cummax()     drawdown = cum /
rolling_max - 1     plt.plot(drawdown)
    plt.title("Drawdown over time")
plt.show() except Exception as e:
print("Drawdown plot error:", e)

# 4.15 Feature clustering (fixed) try:     from
scipy.cluster.hierarchy import linkage, dendrogram     from
scipy.spatial.distance import squareform

    corr = train_ffill[num_cols].corr()
dist = 1 - corr
    dist_condensed = squareform(dist.values, checks=False)
    link = linkage(dist_condensed, method='ward')
dendrogram(link, labels=num_cols, leaf_rotation=90)
plt.title("Hierarchical clustering of features")
plt.show()

    sns.clustermap(corr, cmap="coolwarm", figsize=(10, 10))
plt.title("Feature correlation clustermap")     plt.show()
except Exception as e:     print("Feature clustering
error:", e)
```

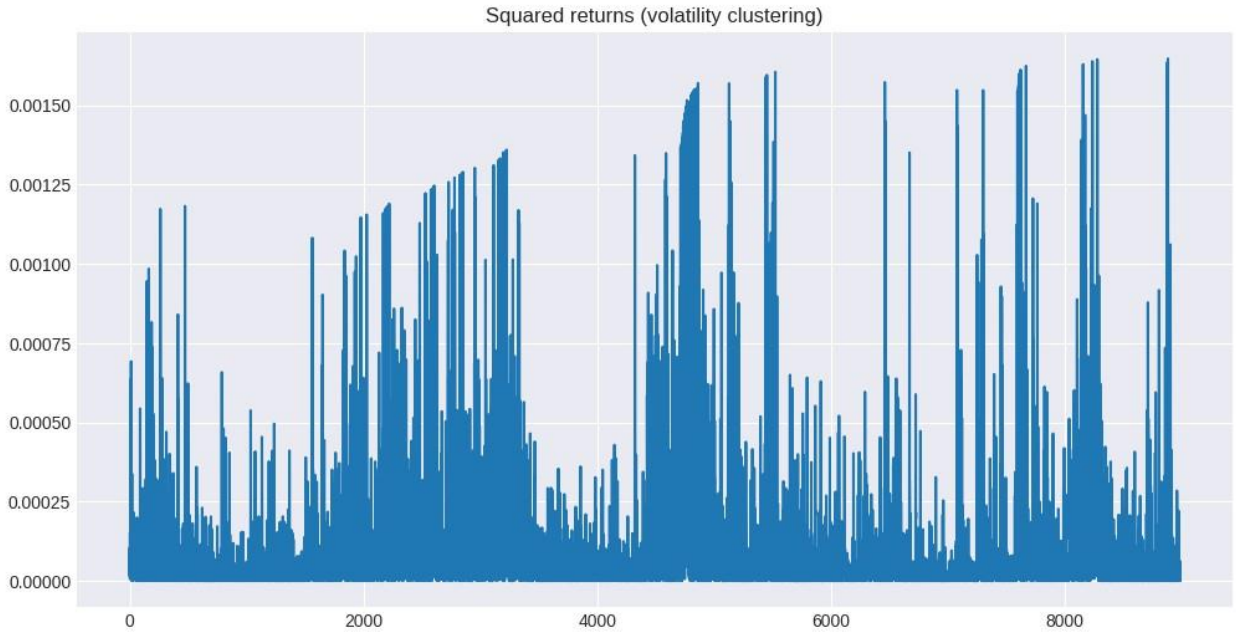
```
# 4.16 Rolling beta vs. risk-free rate if
"risk_free_rate" in train_ffill.columns:
try:         roll_beta =
train_ffill[TARGET].rolling(252).cov(train_ffill["risk_free_rate"]) /
\
                    train_ffill["risk_free_rate"].rolling(252).var()
roll_beta.plot()
        plt.title("Rolling beta vs. risk-free rate")
plt.show()     except Exception as e:
print("Rolling beta error:", e)

# 4.17 Heatmap of rolling correlations (multi-feature) — robust
try:     subset = num_cols[:10]     roll_corr =
train_ffill[subset].rolling(252).corr().groupby(level=0).mean()
    # replace inf/NaN with 0
    roll_corr.replace([np.inf, -np.inf], np.nan, inplace=True)
roll_corr.fillna(0, inplace=True)     sns.heatmap(roll_corr,
cmap="coolwarm")
    plt.title("Average rolling 252-day correlations (subset)")
plt.show() except Exception as e:     print("Rolling
correlation heatmap error:", e)

# 4.18 PCA 2D scatter try:     pc_df = pd.DataFrame(pc_comp[:,
:2], columns=["PC1","PC2"])     pc_df['target'] =
train_ffill[TARGET].values
    plt.scatter(pc_df["PC1"], pc_df["PC2"], c=pc_df["target"],
cmap="coolwarm", alpha=0.5)
    plt.title("PCA 2D scatter colored by target")
plt.colorbar(label="Target")     plt.show()
except Exception as e:     print("PCA scatter
error:", e)

# 4.19 Feature volatility (rolling std of features) try:
feat_vol = train_ffill[num_cols].rolling(252).std().mean()
feat_vol.sort_values(ascending=False).head(20).plot(kind='bar')
plt.title("Top 20 volatile features")     plt.show() except
Exception as e:     print("Feature volatility error:", e)

# 4.20 Outlier count per feature
```

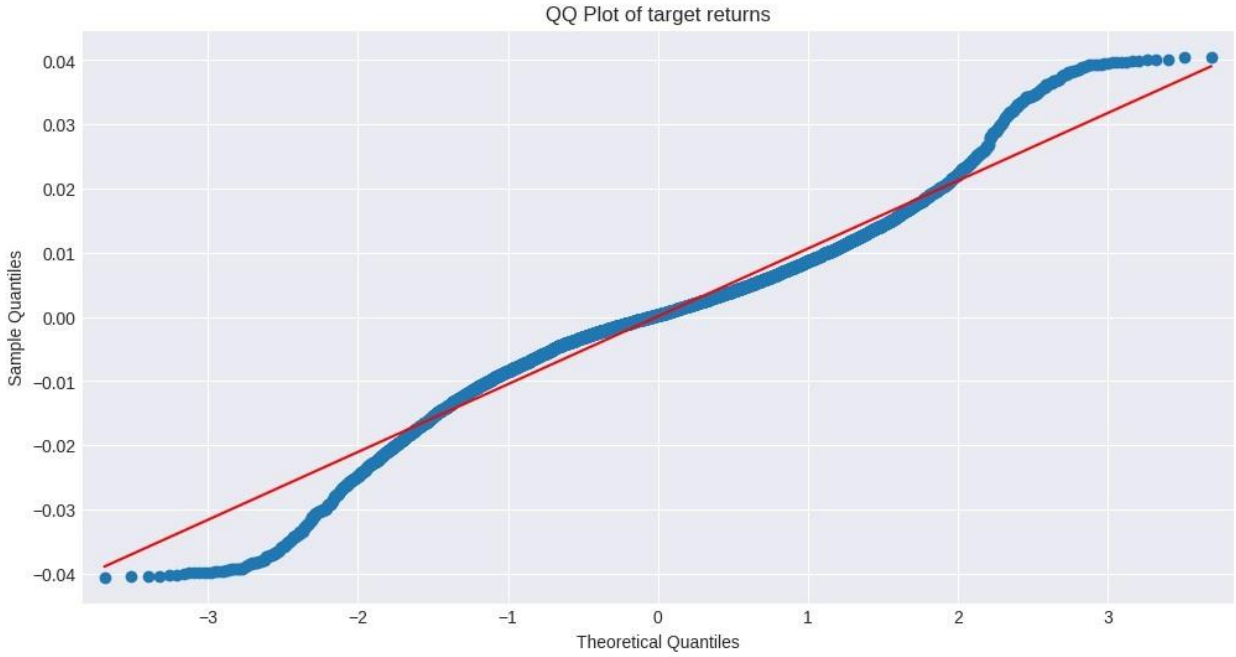
```
try:                zscores    =    stats.zscore(train_ffill[num_cols],
nan_policy='omit')      outlier_counts = (np.abs(zscores) > 3).sum()

outlier_counts.sort_values(ascending=False).head(20).plot(kind='bar')
plt.title("Outlier counts per feature")      plt.show() except
Exception as e:      print("Outlier counts error:", e)

print("\n Advanced EDA: 20+ robust analyses complete.")
```

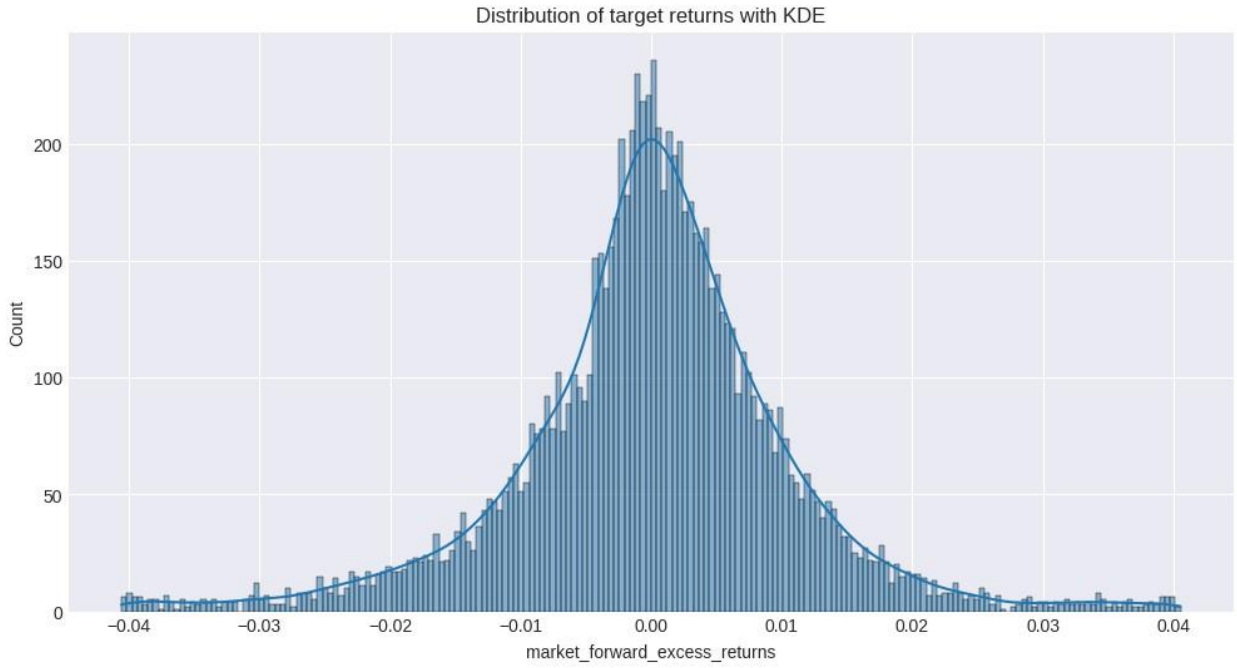



Partial Autocorrelation of target returns
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
0
10
20
30
40
50
60

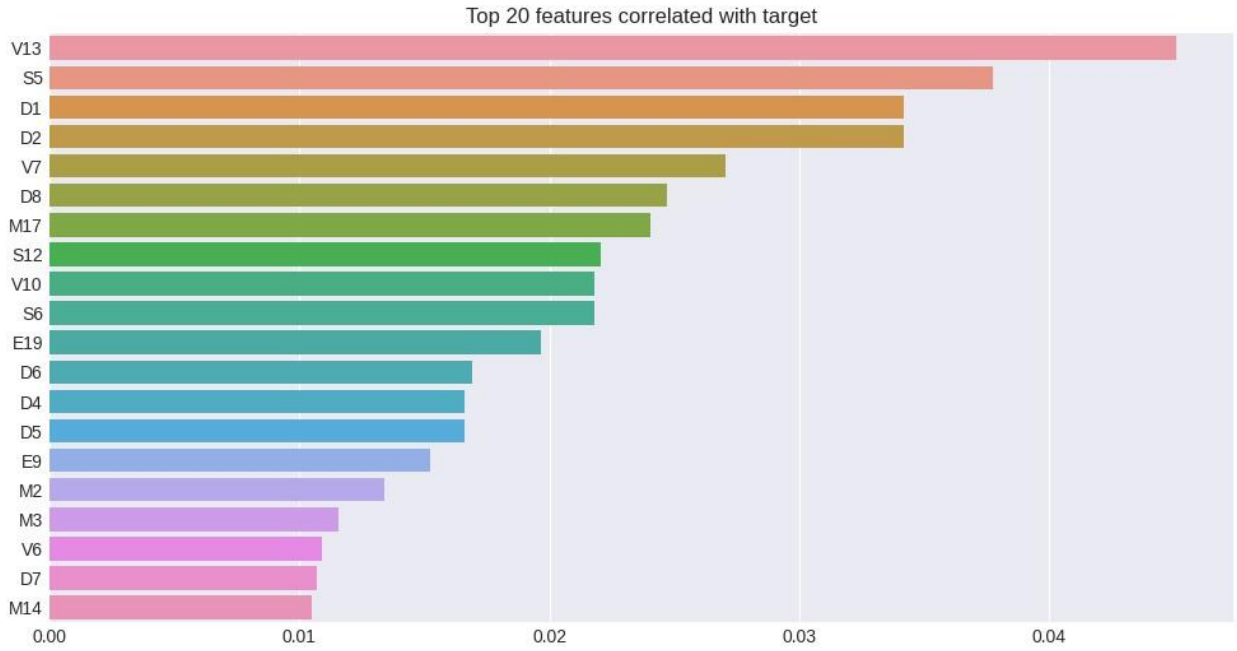

252-day rolling Sharpe ratio
0.15
0.10
0.05
0.00
−0.05
−0.10
−0.15
0
2000
4000
6000
8000

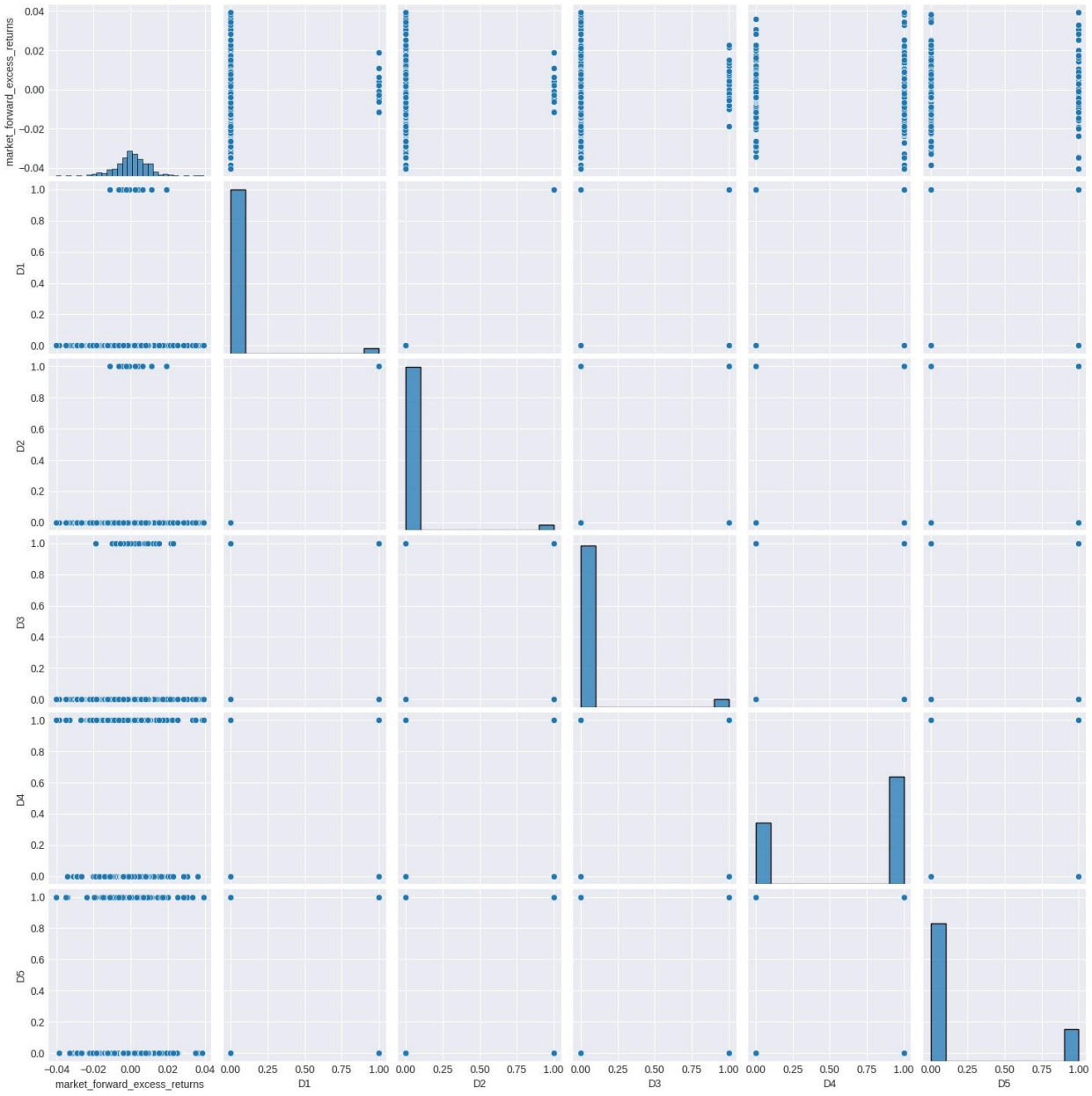

Squared returns (volatility clustering)
0.00150
0.00125
0.00100
0.00075
0.00050
0.00025
0.00000
0
2000
4000
6000
8000

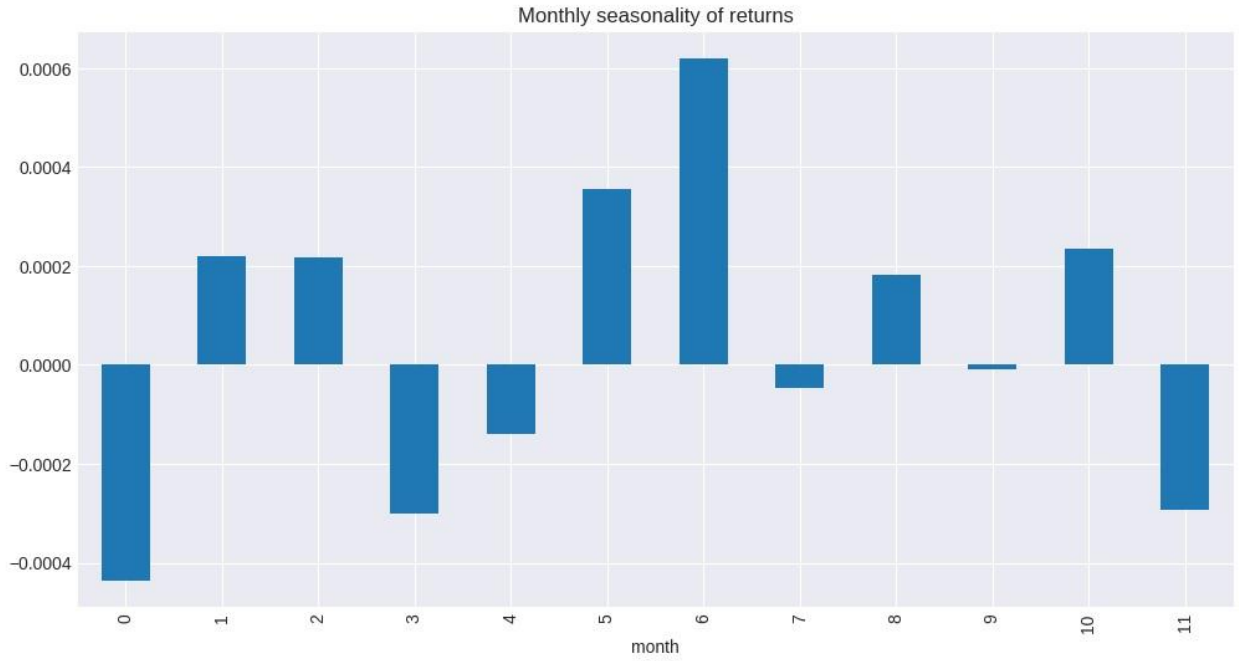

QQ Plot of target returns
Sample Quantiles
0.04
0.03
0.02
0.01
0.00
−0.01
−0.02
−0.03
−0.04
−3
−2
−1
0
1
2
3
Theoretical Quantiles

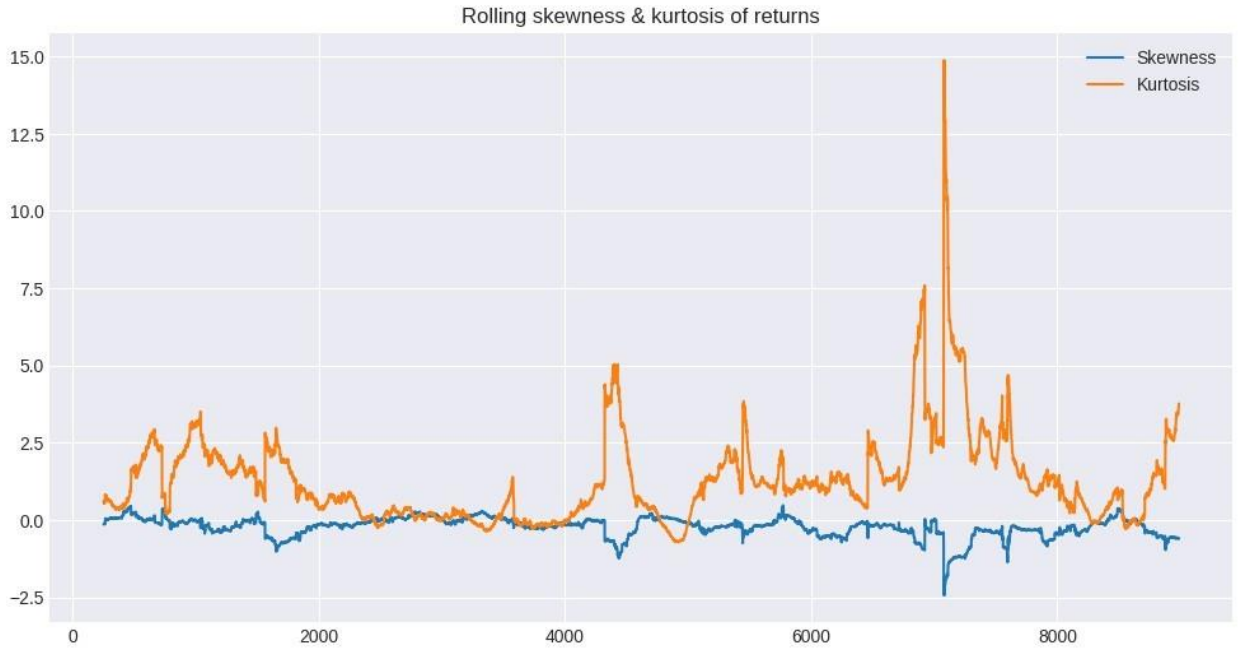


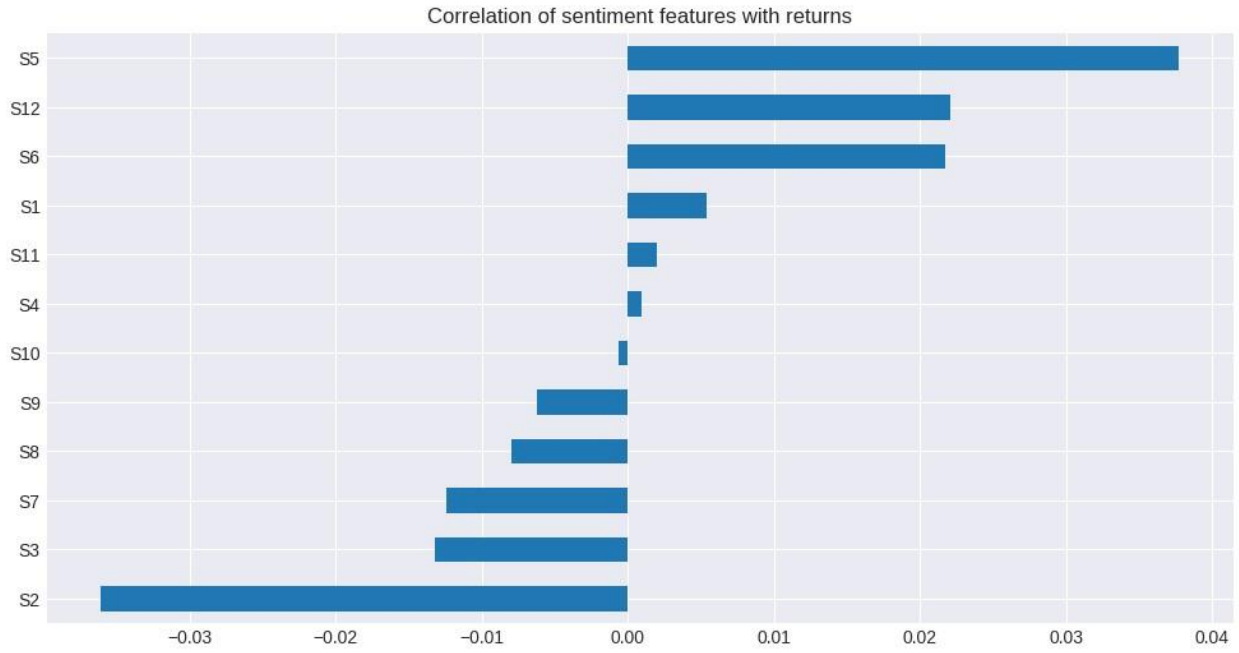

```
Lag-1 correlation with target: -0.0447
Lag-5 correlation with target: -0.0236
Lag-21 correlation with target: -0.0033
Lag-63 correlation with target: 0.0089
```

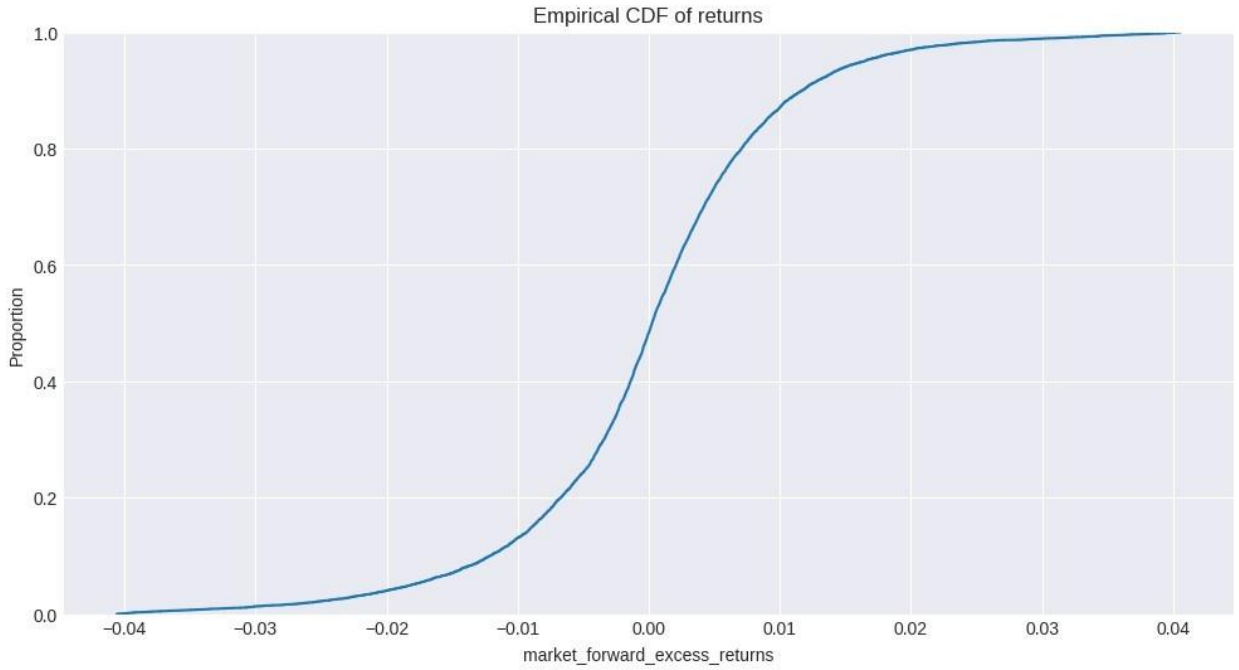

Top 20 features correlated with target
V13
S5
D1
D2
V7
D8
M17
S12
V10
S6
E19
D6
D4
D5
E9
M2
M3
V6
D7
M14
0.00
0.01
0.02
0.03
0.04

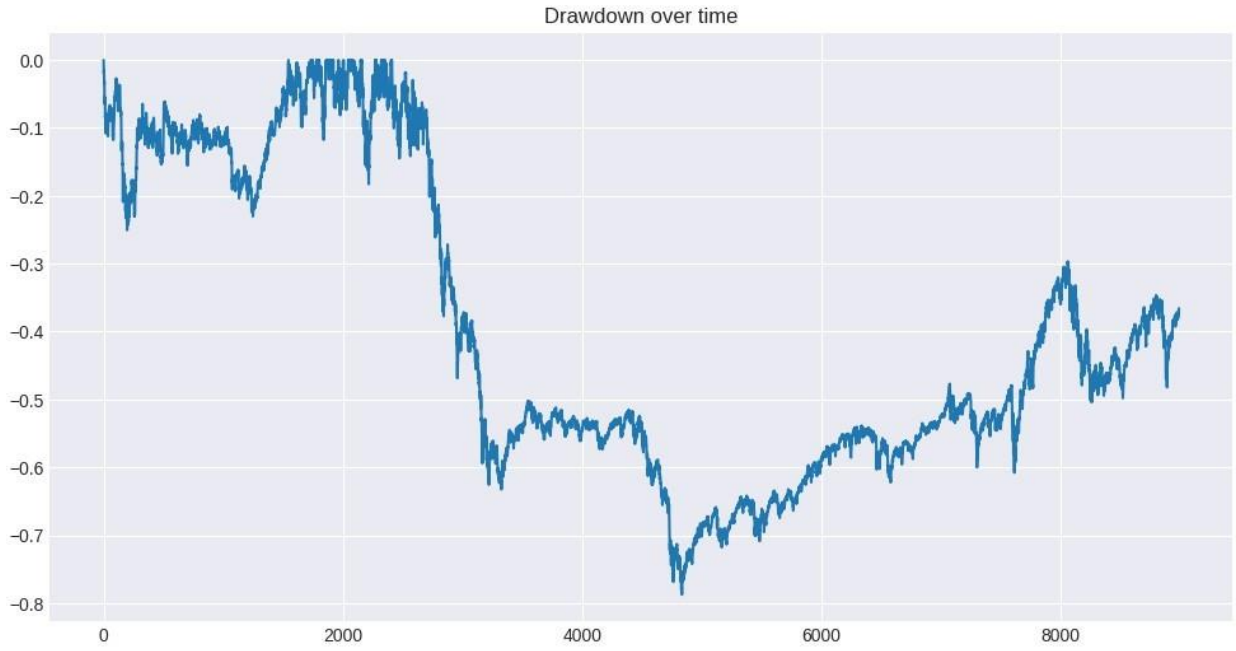

market_forward_excess_returns
D1
D2
D3
D4
D5
0.04
0.02
0.00
−0.02
−0.04
1.0
0.8
0.6
0.4
0.2
0.0
0.00
0.25
0.50
0.75
1.00

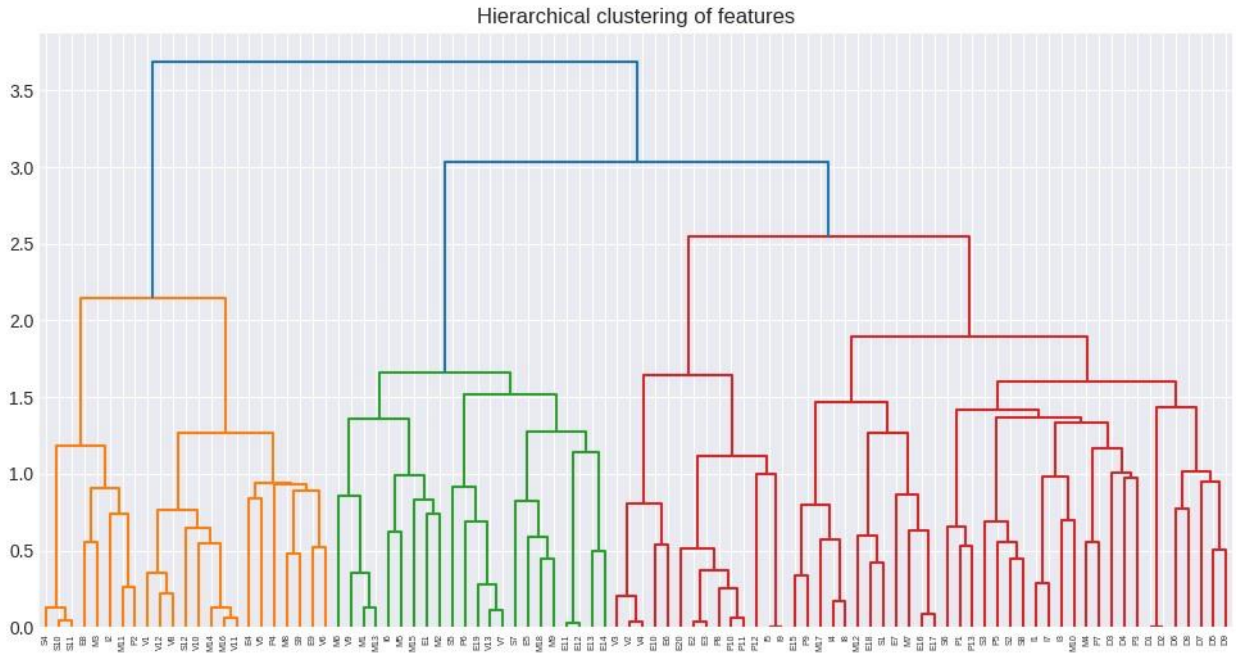

Monthly seasonality of returns
0.0006
0.0004
0.0002
0.0000
−0.0002
−0.0004
0
1
2
3
4
5
6
7
8
9
10
11
month

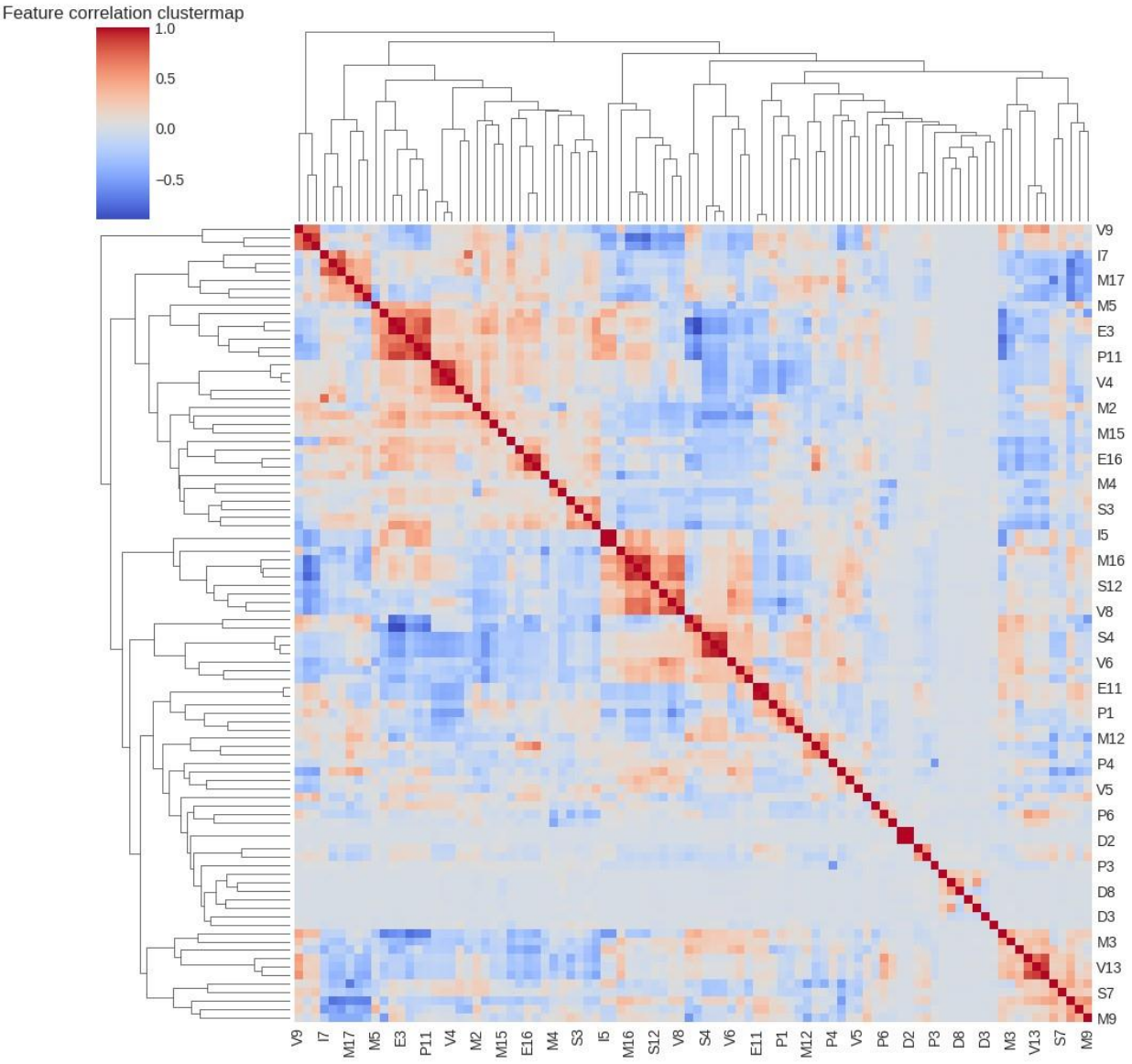

Rolling skewness & kurtosis of returns
Skewness
Kurtosis
15.0
12.5
10.0
7.5
5.0
2.5
0.0
−2.5
0
2000
4000
6000
8000

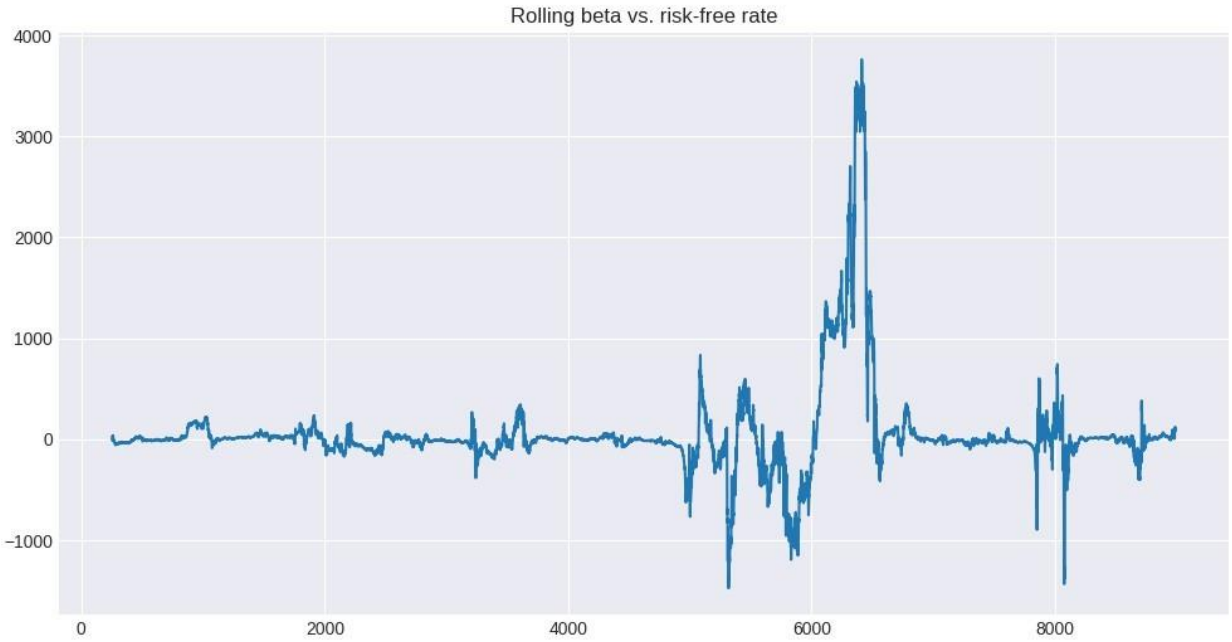

Correlation of sentiment features with returns
S5
S12
S6
S1
S11
S4
S10
S9
S8
S7
S3
S2
−0.03
−0.02
−0.01
0.00
0.01
0.02
0.03
0.04

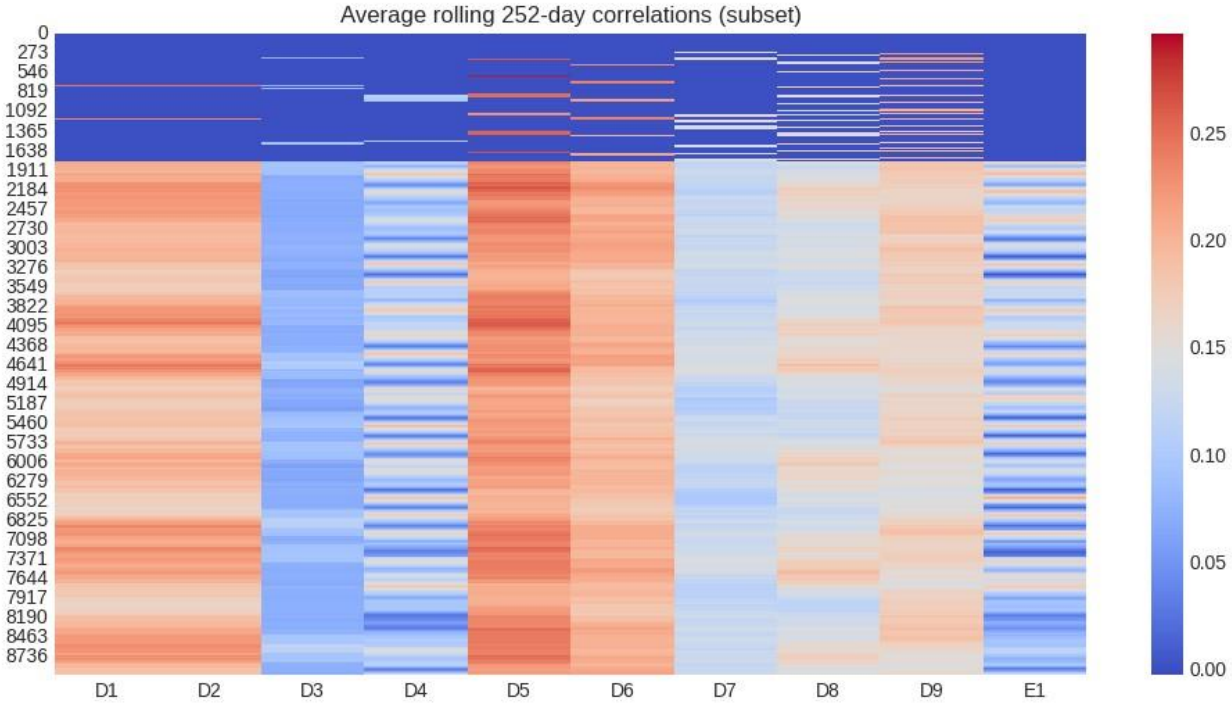

Empirical CDF of returns
Proportion
1.0
0.8
0.6
0.4
0.2
0.0
−0.04
−0.03
−0.02
−0.01
0.00
0.01
0.02
0.03
0.04
market_forward_excess_returns

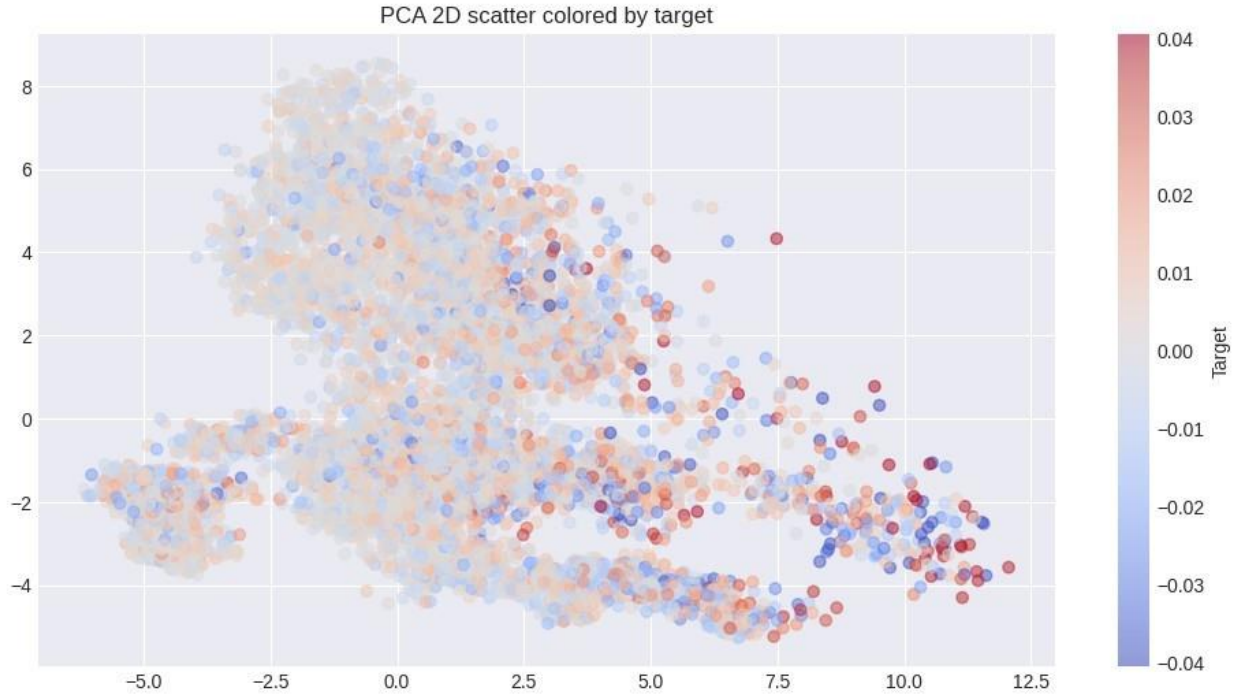

Drawdown over time
0.0
−0.1
−0.2
−0.3
−0.4
−0.5
−0.6
−0.7
−0.8
0
2000
4000
6000
8000

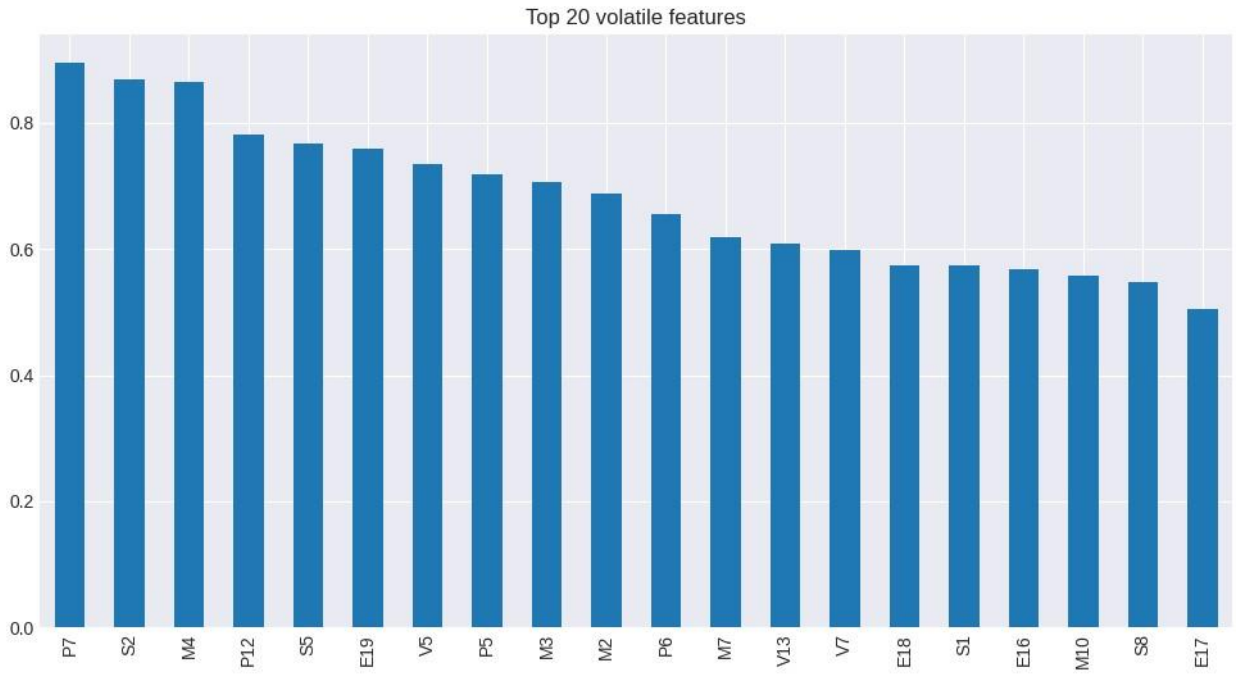

Hierarchical clustering of features
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0


Feature correlation clustermap
1.0
0.5
0.0
−0.5
V9
I7
M17
M5
E3
P11
V4
M2
M15
E16
M4
S3
I5
M16
S12
V8
S4
V6
E11
P1
M12
P4
V5
P6
D2
P3
D8
D3
M3
V13
S7
M9


Rolling beta vs. risk-free rate
4000
3000
2000
1000
0
−1000
0
2000
4000
6000
8000


Average rolling 252-day correlations (subset)
0
273
546
819
1092
1365
1638
1911
2184
2457
2730
3003
3276
3549
3822
4095
4368
4641
4914
5187
5460
5733
6006
6279
6552
6825
7098
7371
7644
7917
8190
8463
8736
D1
D2
D3
D4
D5
D6
D7
D8
D9
E1
0.25
0.20
0.15
0.10
0.05
0.00






```
Outlier counts error: 'numpy.int64' object has no attribute
'sort_values'

 Advanced EDA: 20+ robust analyses complete.
!pip install hurst
```

```
Collecting hurst
  Downloading hurst-0.0.5-py3-none-any.whl.metadata (3.6 kB)
Requirement already satisfied: pandas>=0.18 in
/usr/local/lib/python3.11/dist-packages (from hurst) (2.2.3)
Requirement already satisfied: numpy>=1.10 in
/usr/local/lib/python3.11/dist-packages (from hurst) (1.26.4)
Requirement already satisfied: mkl_fft i
/usr/local/lib/python3.11/dist-packages (from numpy>=1.10->hurst)
(1.3.8)
Requirement already satisfied: mkl_random i
/usr/local/lib/python3.11/dist-packages (from numpy>=1.10->hurst)
(1.2.4)
Requirement already satisfied: mkl_umath in
/usr/local/lib/python3.11/dist-packages (from numpy>=1.10->hurst)
(0.1.1)
Requirement already satisfied: mkl in /usr/local/lib/python3.11/dist-
packages (from numpy>=1.10->hurst) (2025.2.0)
Requirement already satisfied: tbb4py i
/usr/local/lib/python3.11/dist-packages (from numpy>=1.10->hurst)
(2022.2.0)
Requirement already satisfied: mkl-service in
/usr/local/lib/python3.11/dist-packages (from numpy>=1.10->hurst)
(2.4.1)
Requirement already satisfied: python-dateutil>=2.8.2 in
/usr/local/lib/python3.11/dist-packages (from pandas>=0.18->hurst)
(2.9.0.post0)
Requirement already satisfied: pytz>=2020.1 in
/usr/local/lib/python3.11/dist-packages (from pandas>=0.18->hurst)
(2025.2)
Requirement already satisfied: tzdata>=2022.7 i
/usr/local/lib/python3.11/dist-packages (from pandas>=0.18->hurst)
(2025.2)
Requirement already satisfied: six>=1.5 in
/usr/local/lib/python3.11/dist-packages (from python-dateutil>=2.8.2-
>pandas>=0.18->hurst) (1.17.0)
Requirement already satisfied: intel-openmp<2026,>=2024 in
/usr/local/lib/python3.11/dist-packages (from mkl->numpy>=1.10->hurst)
(2024.2.0)
Requirement already satisfied: tbb==2022.* in
```

```
/usr/local/lib/python3.11/dist-packages (from mkl->numpy>=1.10->hurst)
(2022.2.0)

Requirement already satisfied: tcmlib==1.* in
/usr/local/lib/python3.11/dist-packages (from tbb==2022.*->mkl-
>numpy>=1.10->hurst) (1.4.0)

Requirement already satisfied: intel-cmplr-lib-rt in
/usr/local/lib/python3.11/dist-packages (from mkl_umath->numpy>=1.10-
>hurst) (2024.2.0)

Requirement already satisfied: intel-cmplr-lib-ur==2024.2.0 in
/usr/local/lib/python3.11/dist-packages (from intel-

openmp<2026,>=2024->mkl->numpy>=1.10->hurst) (2024.2.0)
```

```
Downloading hurst-0.0.5-py3-none-any.whl (5.9 kB)
Installing collected packages: hurst
Successfully installed hurst-0.0.5

# ------------------------
# 5) More Advanced EDA (20+)
# ------------------------
import numpy as np import
pandas as pd
import matplotlib.pyplot as plt
import statsmodels.api as sm from
scipy.signal import welch from
scipy.stats import entropy from
itertools import combinations from
scipy import stats
import statsmodels.formula.api as smf

# Helper: robust Hurst exponent via R/S (fallback)
def compute_hurst_rs(series, min_window=10, max_window=None,
num_sizes=20):
    """
    Compute Hurst exponent using Rescaled Range (R/S) method.
    Returns H (float) or np.nan if not computable.
    """
    s = np.asarray(series)
s = s[~np.isnan(s)]     n
= len(s)     if n < 2 *
min_window:
        return np.nan

    if max_window is None:
max_window = n // 2     if
max_window <= min_window:
return np.nan

    # window sizes on log scale, unique, integer
window_sizes =
np.unique(np.floor(np.logspace(np.log10(min_window),
np.log10(max_window), num=num_sizes)).astype(int))
rs_vals = []     sizes = []     for w in
window_sizes:         if w < 2:             continue
num_segments = n // w         if num_segments < 2:
continue         seg_rs = []         for i in
range(num_segments):
            seg = s[i*w:(i+1)*w]
```

```
            if seg.size < 2 or np.nanstd(seg) == 0:
continue
            mean_adj = seg - np.nanmean(seg)
Y = np.cumsum(mean_adj)
            R = np.nanmax(Y) - np.nanmin(Y)
S = np.nanstd(seg)              if S <= 0:
continue              seg_rs.append(R / S)
if len(seg_rs) == 0:              continue
        rs_mean = np.nanmean(seg_rs)
if rs_mean <= 0:
            continue
        rs_vals.append(rs_mean)
sizes.append(w)

    if len(sizes) < 3 or len(rs_vals) < 3:
return np.nan

    # linear regression on log-log     try:
slope, intercept = np.polyfit(np.log10(sizes),
np.log10(rs_vals), 1)         H = slope         return
float(H)     except Exception:         return np.nan

# 5.1 Value at Risk (VaR) & Conditional VaR
alpha = 0.05
VaR = train_ffill[TARGET].quantile(alpha)
CVaR = train_ffill[TARGET][train_ffill[TARGET] < VaR].mean()
print(f"5% VaR: {VaR:.6f}, CVaR: {CVaR:.6f}")

# 5.2 Rolling VaR
roll_VaR = train_ffill[TARGET].rolling(252).quantile(0.05)
plt.plot(roll_VaR); plt.title("Rolling 5% VaR"); plt.show()

# 5.3 Max drawdown duration
# ensure 'drawdown' exists (from prior section); if not, compute
try:     dd = drawdown.copy() except NameError:     cum = (1 +
train_ffill[TARGET].fillna(0)).cumprod()     rolling_max =
cum.cummax()     dd = cum / rolling_max - 1
# compute duration of current drawdown run (days since peak) dd_dur =
(dd == 0).astype(int).groupby((dd == 0).cumsum()).cumcount()
plt.plot(dd_dur); plt.title("Drawdown duration"); plt.show()
```

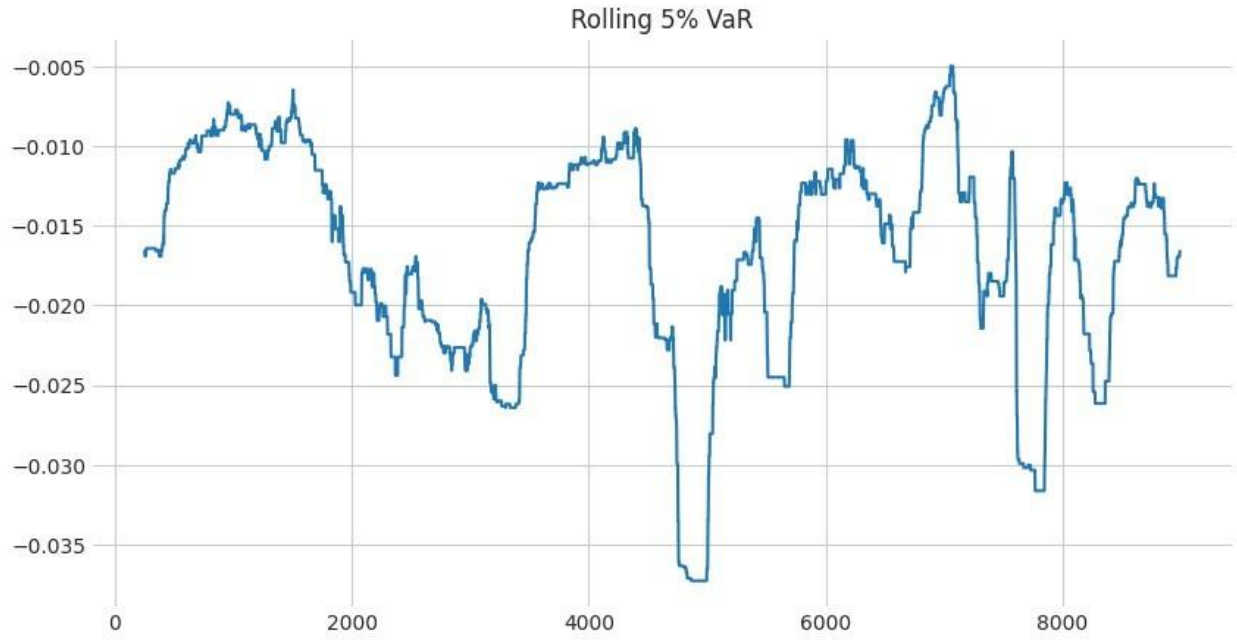
```
# 5.4 Extreme tail dependence (upper vs lower quantiles)
low_tail = train_ffill[TARGET][train_ffill[TARGET] <
train_ffill[TARGET].quantile(0.05)]
high_tail = train_ffill[TARGET][train_ffill[TARGET] >
train_ffill[TARGET].quantile(0.95)]
print("Mean lower 5%:", float(low_tail.mean()), "| Mean upper 5%:",
float(high_tail.mean()))

# 5.5 Hurst exponent (mean reversion vs trending) — robust (try hurst,
fallback to RS)
series_for_hurst = train_ffill[TARGET].dropna().values H_value =
np.nan if len(series_for_hurst) < 50 or
np.nanstd(series_for_hurst) == 0:     print("Hurst: series too
short or zero variance — skipping
(returned NaN).")
else:     try:
        # try using hurst package if available
        from hurst import compute_Hc
        H, c, val = compute_Hc(series_for_hurst, kind='price')
H_value = float(H)
        print("Hurst exponent (hurst lib):", H_value)
except Exception as e:
        # fallback to R/S implementation
        H_fallback = compute_hurst_rs(series_for_hurst, min_window=10,
max_window=None, num_sizes=20)         H_value = H_fallback
        print("Hurst exponent (fallback R/S):", H_value, "| reason:",
str(e))

# 5.6 Spectral density of returns try:     f, Pxx =
welch(train_ffill[TARGET].dropna(), nperseg=min(256, max(16,
len(train_ffill[TARGET].dropna()))))
    plt.semilogy(f, Pxx); plt.title("Spectral density of returns");
plt.xlabel("Frequency"); plt.ylabel("Power"); plt.show() except
Exception as e:     print("Spectral density error:", e)

# 5.7 Rolling entropy (uncertainty of returns) try:
roll_entropy = train_ffill[TARGET].rolling(252).apply(lambda x:
entropy(np.histogram(x, bins=20)[0]+1), raw=True)
    plt.plot(roll_entropy); plt.title("Rolling entropy of returns");
plt.show() except Exception as e:     print("Rolling entropy
error:", e)

# 5.8 Cross-correlation with sentiment index
```

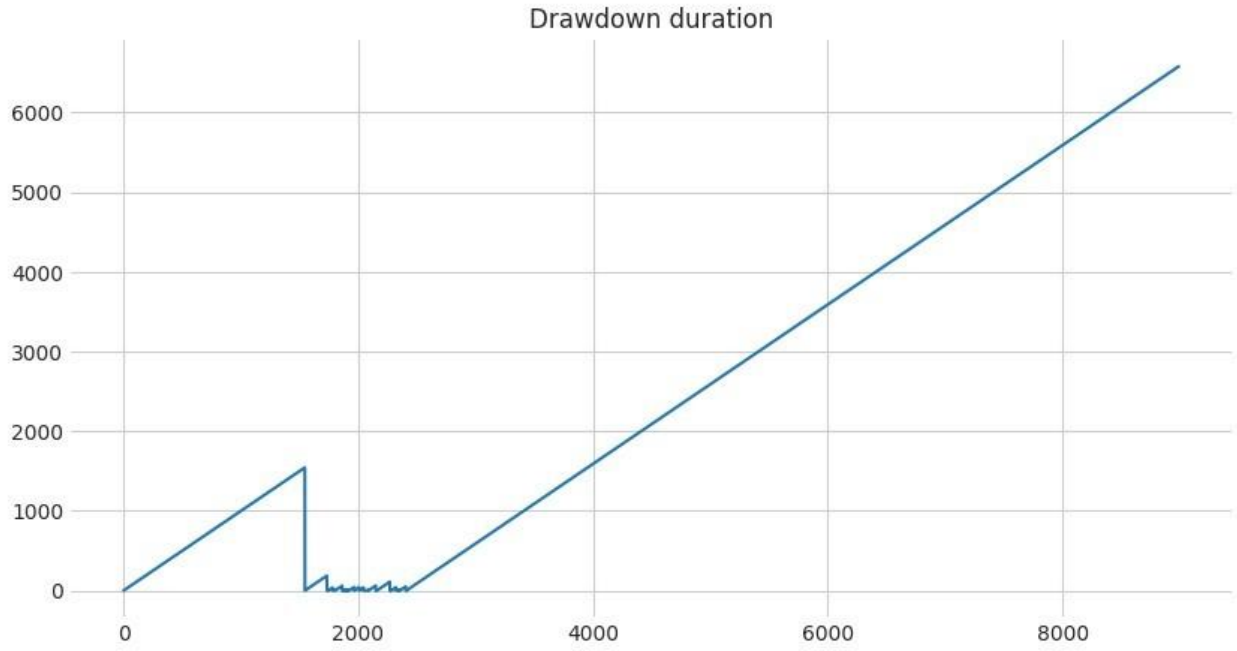
```
if 'sentiment_cols' in globals() and sentiment_cols:
lead_corrs = {}     for lag in [0,1,5,21]:
lead_corrs[lag] =
train_ffill[TARGET].corr(train_ffill[sentiment_cols[0]].shift(lag))
print("Lagged correlations (sentiment vs returns):", lead_corrs)

# 5.9 Feature-target mutual information try:     from
sklearn.feature_selection import mutual_info_regression     mi =
mutual_info_regression(train_ffill[num_cols].fillna(0),
train_ffill[TARGET].fillna(0))
mi_series = pd.Series(mi,
index=num_cols).sort_values(ascending=False)
    mi_series.head(20).plot(kind='bar'); plt.title("Top 20 features by
Mutual Information"); plt.show() except Exception as e:
print("Mutual information error:", e)

# 5.10 Rolling feature importance (correlation windowed)
try:
    # compute rolling correlation of each feature with target; this
returns MultiIndex (index, column)     corr_roll =
train_ffill[num_cols].rolling(252).corr(train_ffill[TARGET])
# average recent correlations by feature and plot tail
recent = corr_roll.groupby(level=0).mean().tail(50).T
    recent.plot(legend=False)
    plt.title("Recent rolling correlations with target"); plt.show()
except Exception as e:     print("Rolling feature correlation
error:", e)

# 5.11 Cointegration with interest rates if "I1" in
train_ffill.columns:     try:         from
statsmodels.tsa.stattools import coint         idx =
train_ffill[[TARGET, "I1"]].dropna().index         t1 =
train_ffill.loc[idx, TARGET]         t2 =
train_ffill.loc[idx, "I1"]         score, pvalue, _ =
coint(t1, t2)
        print("Cointegration test (Target vs I1): p-value =", pvalue)
except Exception as e:         print("Cointegration error:", e)

# 5.12 Rolling volatility ratio (short-term / long-term)
short_vol = train_ffill[TARGET].rolling(21).std()
long_vol = train_ffill[TARGET].rolling(252).std()
vol_ratio = (short_vol/long_vol)
vol_ratio.plot()
plt.title("Volatility ratio (21d / 252d)"); plt.show()
```

```
# 5.13 Extreme move clustering (runs of >2σ)
sigma = train_ffill[TARGET].std()
extreme_moves = (np.abs(train_ffill[TARGET]) > 2*sigma).astype(int)
plt.plot(extreme_moves.rolling(63).sum())
plt.title("Frequency of >2σ moves (63d window)"); plt.show()

# 5.14 Structural break test (CUSUM-like) try:     from
statsmodels.stats.diagnostic import breaks_cusumolsresid
    y = train_ffill[TARGET].dropna().values
X = sm.add_constant(np.arange(len(y)))
ols = sm.OLS(y, X).fit()
    stat, pval, crit = breaks_cusumolsresid(ols.resid)
print("CUSUM test p-value:", pval) except Exception as
e:     print("CUSUM structural break test error:", e)

# 5.15 Return quantile regression slopes try:     if
num_cols:         qreg = smf.quantreg(f"{TARGET} ~
{num_cols[0]}", train_ffill.dropna()).fit(q=0.05)
        print("Quantile regression coef (5%):", qreg.params)
except Exception as e:     print("Quantile regression
error:", e)

# 5.16 Transition matrix of return signs try:
signs = np.sign(train_ffill[TARGET].dropna())
    trans = pd.crosstab(signs.shift(), signs, normalize="index")
sns.heatmap(trans, annot=True, cmap="Blues");
plt.title("Transition matrix of return signs"); plt.show()
except Exception as e:     print("Transition matrix error:", e)

# 5.17 Lead-lag Granger causality test if "M1" in
train_ffill.columns:     try:         from
statsmodels.tsa.stattools import grangercausalitytests
subset = train_ffill[[TARGET,"M1"]].dropna()         if len(subset)
> 50:             print("Granger causality test (Target <- M1)")
grangercausalitytests(subset, maxlag=5, verbose=False)
else:             print("Not enough data points for Granger
causality.")     except Exception as e:         print("Granger
causality error:", e)

# 5.18 Feature seasonality strength
```

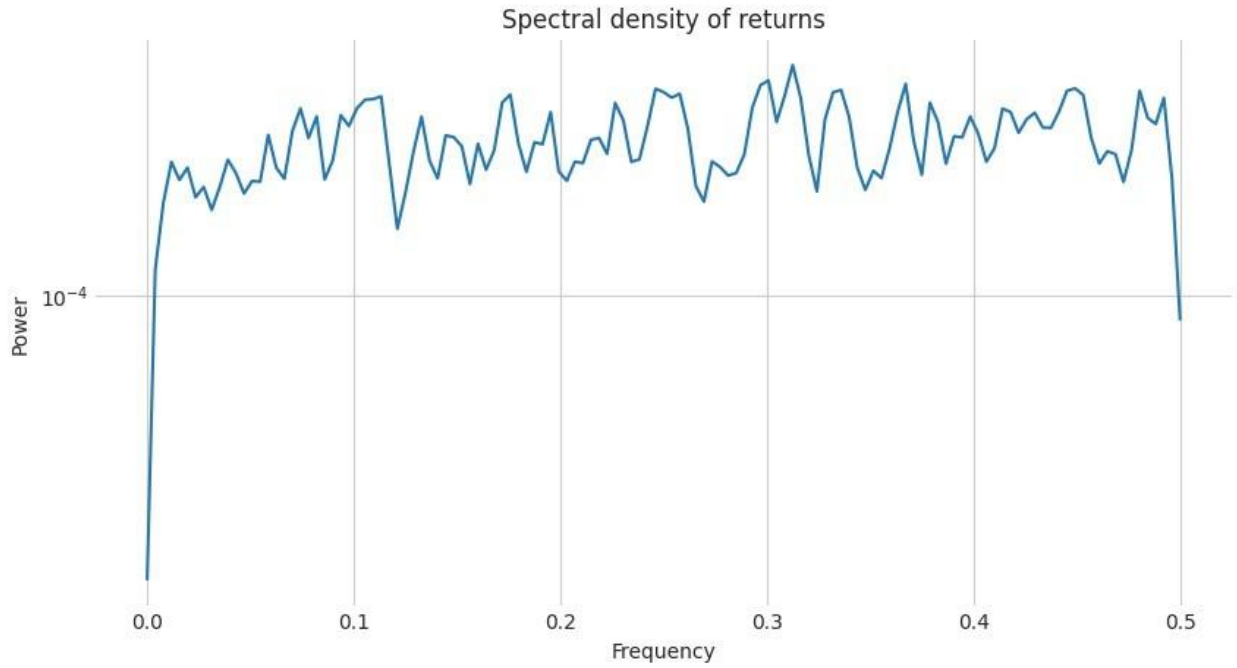
```
try:     if 'month' not in train_ffill.columns:
train_ffill['month'] = train_ffill['date_id'] % 12
season_strength = train_ffill.groupby("month")
[num_cols[:10]].mean().var()
    season_strength.plot(kind="bar"); plt.title("Feature seasonality
strength (subset)"); plt.show() except Exception as e:
print("Seasonality strength error:", e)

# 5.19 Tail dependence heatmap (copula-style approx)
try:     tail_corr = {}     for f1,f2 in
combinations(num_cols[:10],2):         idx =
train_ffill[[f1,f2]].dropna().index         if
len(idx) == 0:
            continue
        joint_prob = np.mean((train_ffill.loc[idx,f1] <
train_ffill.loc[idx,f1].quantile(0.05)) &
                             (train_ffill.loc[idx,f2] <
train_ffill.loc[idx,f2].quantile(0.05)))
tail_corr[(f1,f2)] = joint_prob
    tail_corr_df = pd.DataFrame(list(tail_corr.items()),
columns=["pair","tail_prob"])

tail_corr_df.set_index("pair").tail_prob.nlargest(10).plot(kind="bar")
plt.title("Top 10 feature pairs with strongest lower-tail dependence")
plt.show() except Exception as e:     print("Tail dependence error:",
e)

# 5.20 Calendar effect (day-of-week returns) try:
train_ffill['dow'] = train_ffill['date_id'] % 5     dow_mean
= train_ffill.groupby('dow')[TARGET].mean()
dow_mean.plot(kind="bar"); plt.title("Day-of-week average
returns"); plt.show() except Exception as e:     print("Day-
of-week effect error:", e)

print("\n Added another 20 advanced EDA analyses: VaR/CVaR, entropy,
spectral density, robust Hurst, Granger causality, cointegration,
structural breaks, transition matrices, and more.")
```

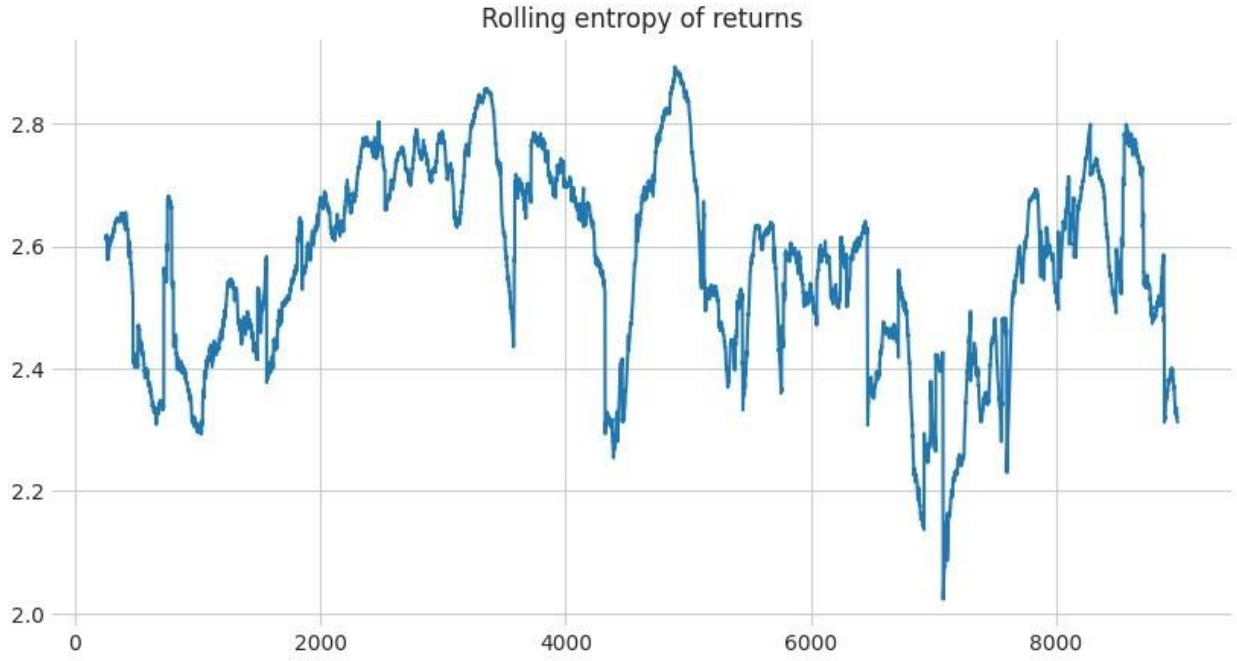
```
5% VaR: -0.017726, CVaR: -0.025440
```



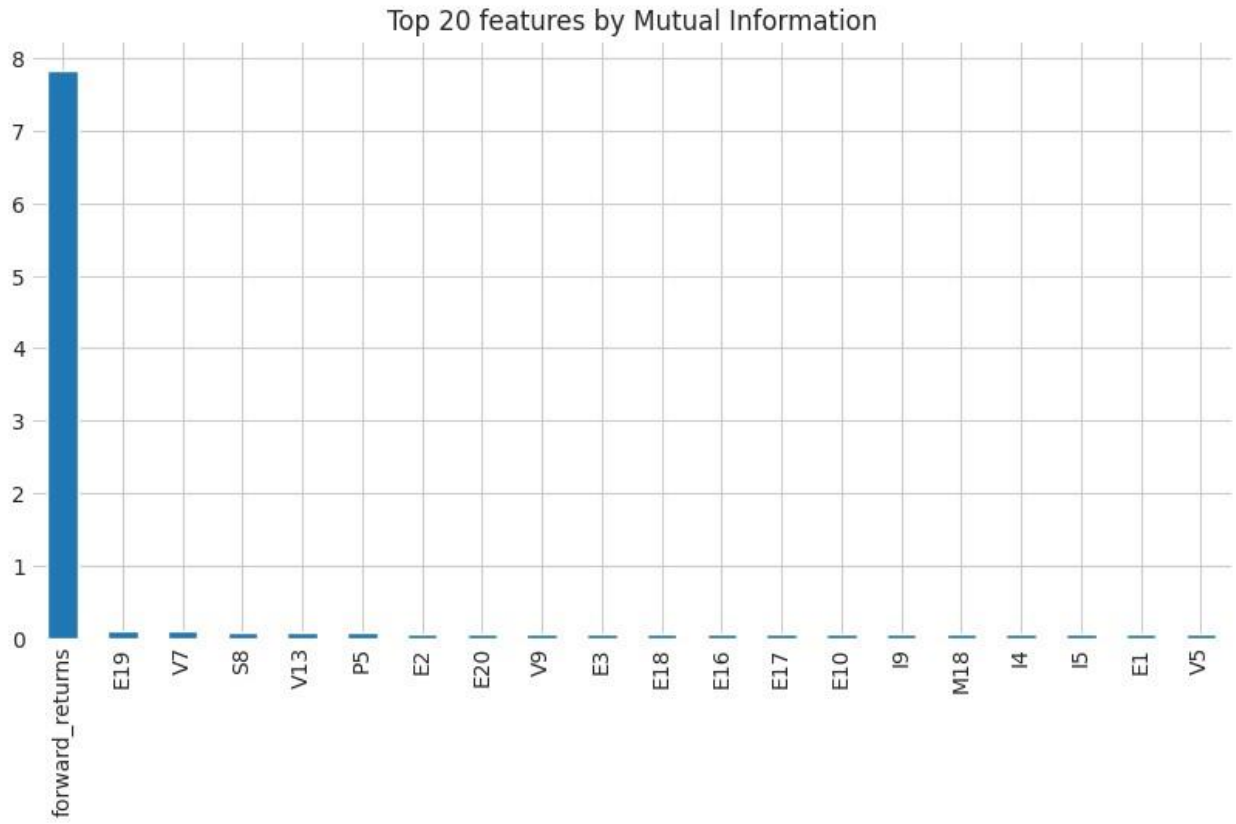


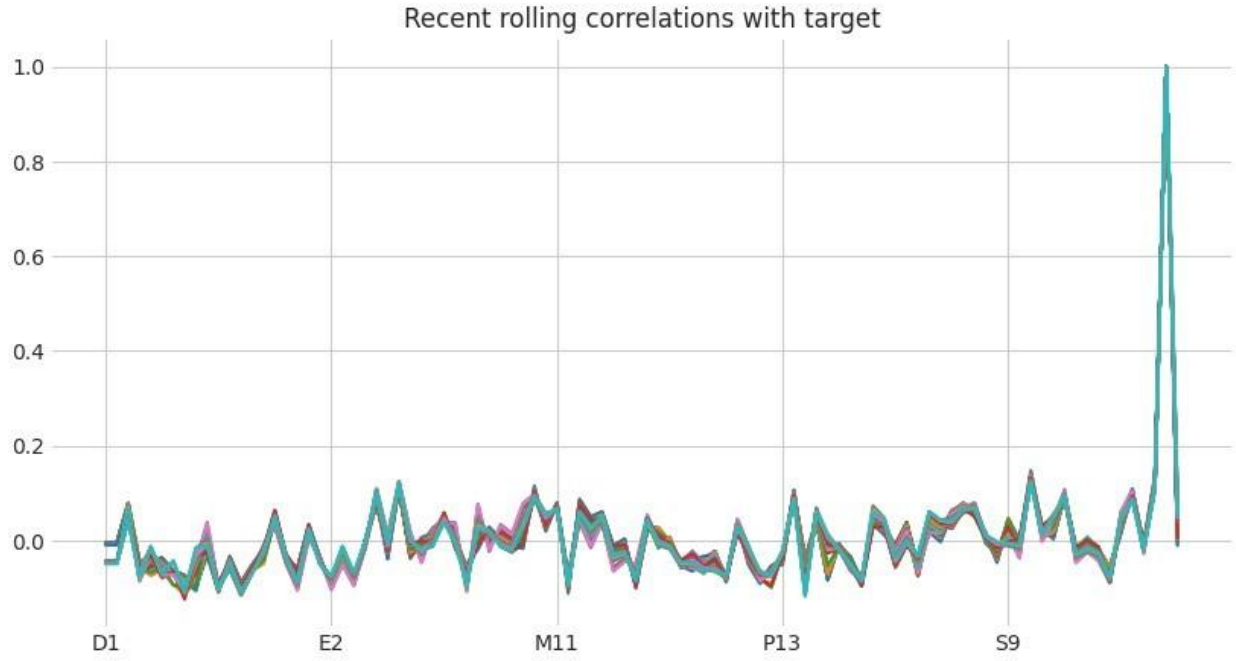

```
Mean lower 5%: -0.02544019979992103 | Mean upper 5%:
0.02435377257790015
Hurst exponent (fallback R/S): 0.5356585333439803 | reason: divide by
zero encountered in divide
```



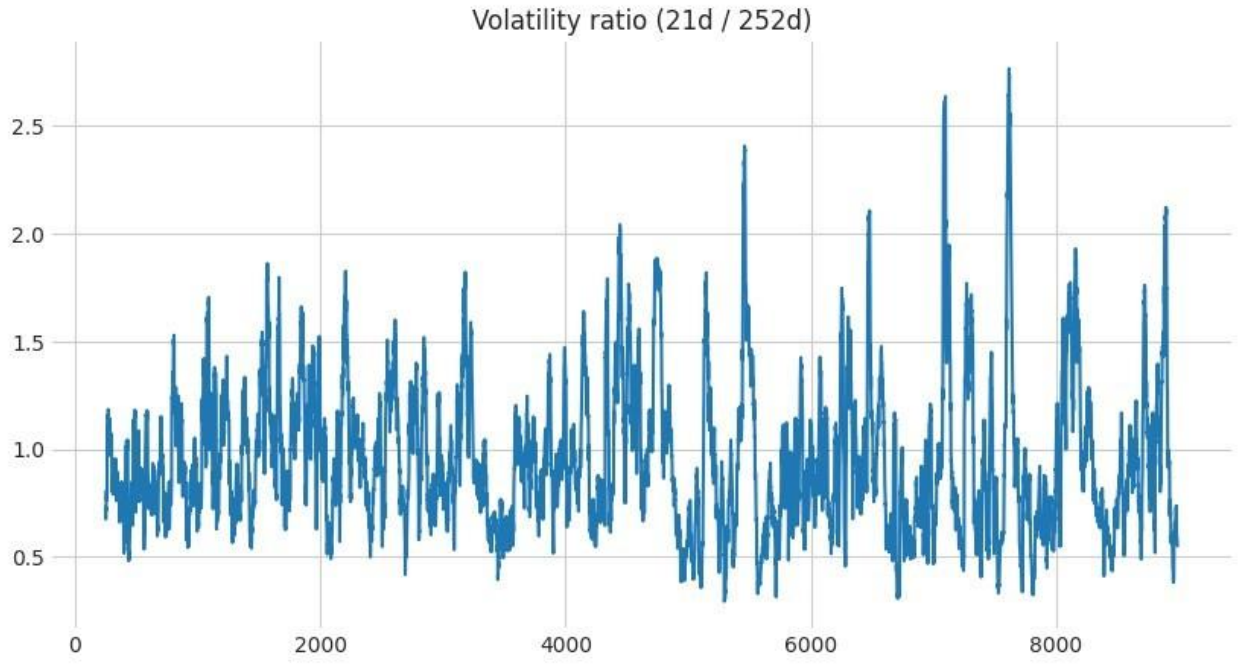


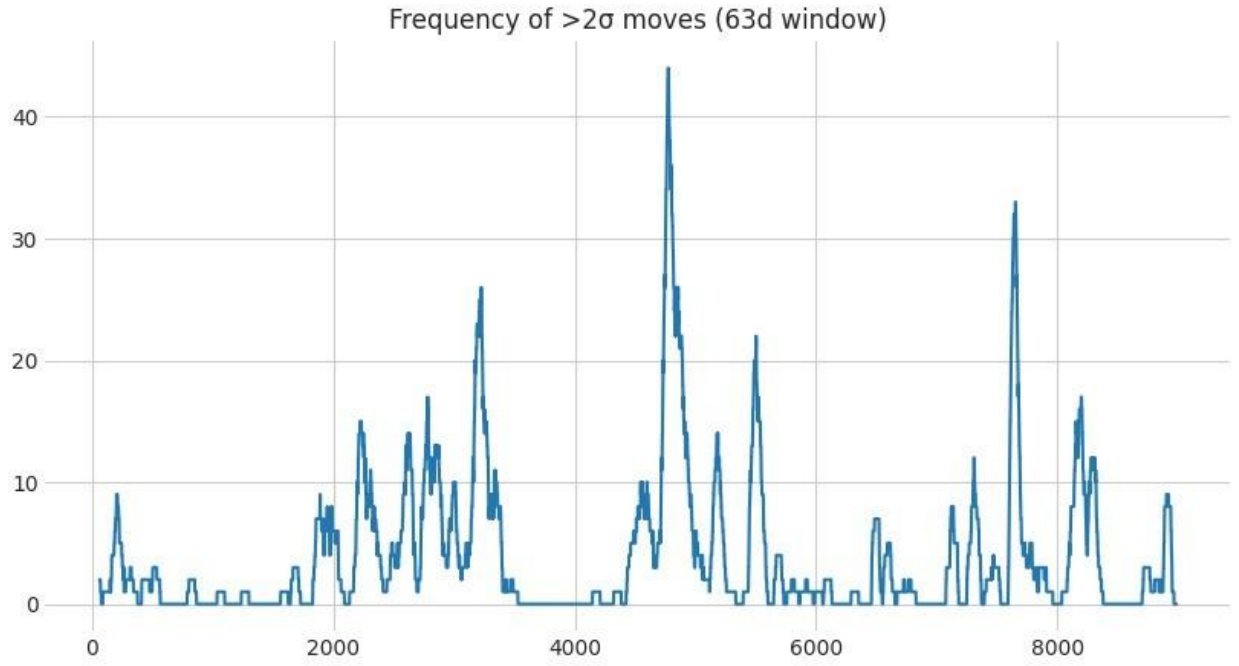

```
Lagged correlations (sentiment vs returns): {0: 0.004823071345147378,
1: 0.004955981572895081, 5: 0.004599778020822951, 21
0.0003496581151897148}
```





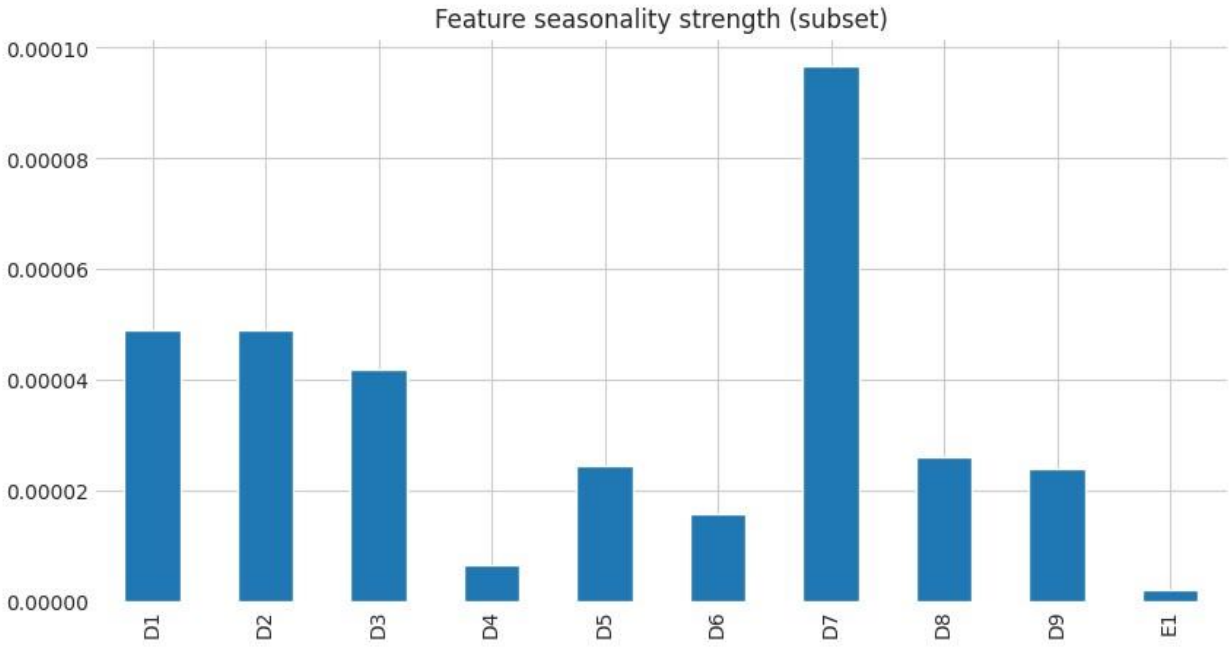

```
Cointegration test (Target vs I1): p-value = 2.9898527136287197e-29
```

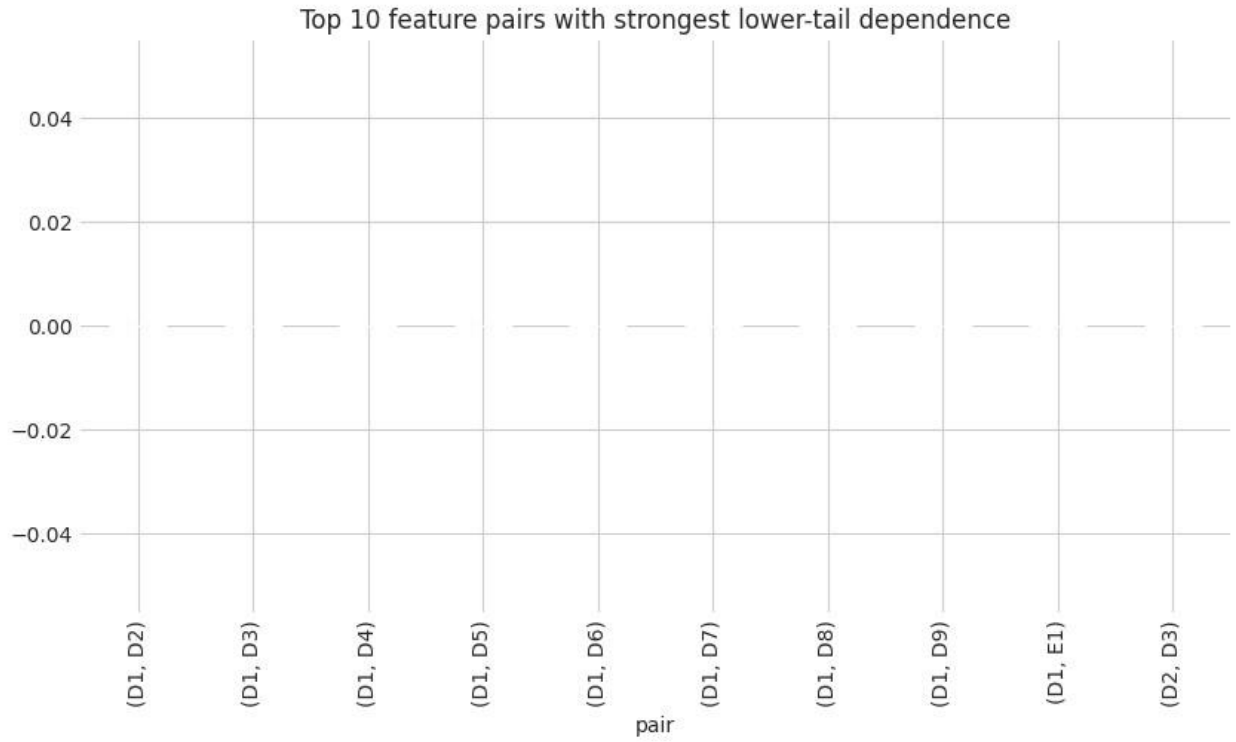


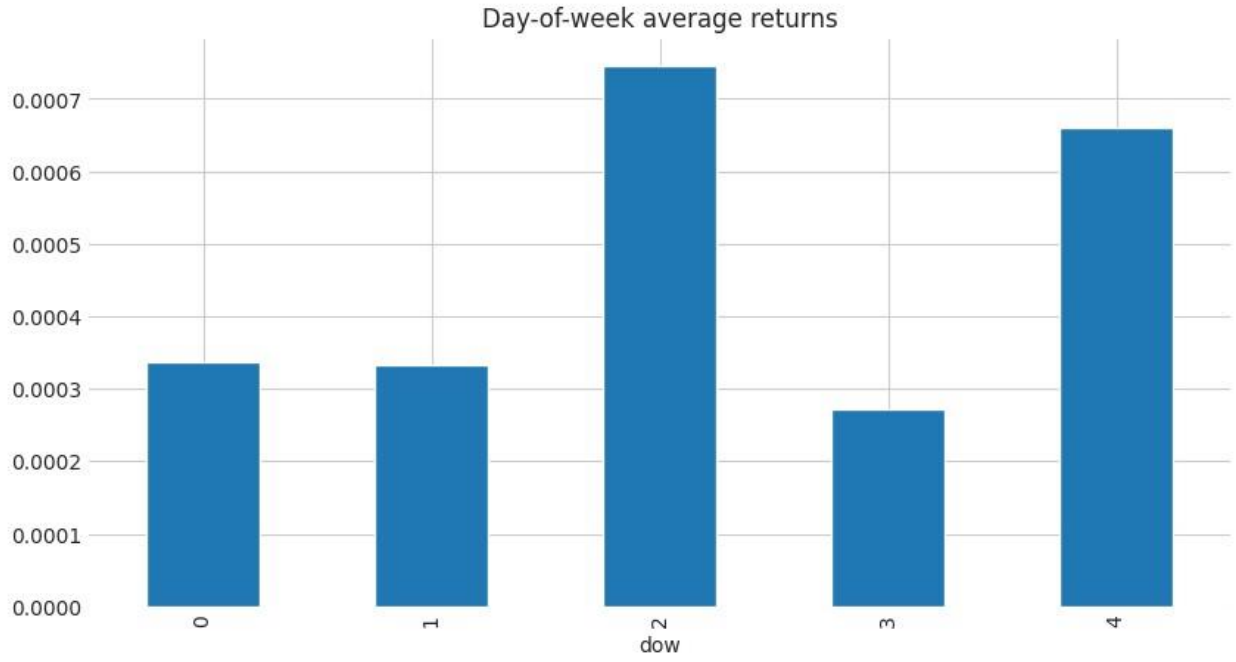



```
CUSUM test p-value: 0.6512456160041034
Quantile regression coef (5%): Intercept   -0.017745
D1           0.000698
dtype: float64
```


```
Granger causality test (Target <- M1)
```

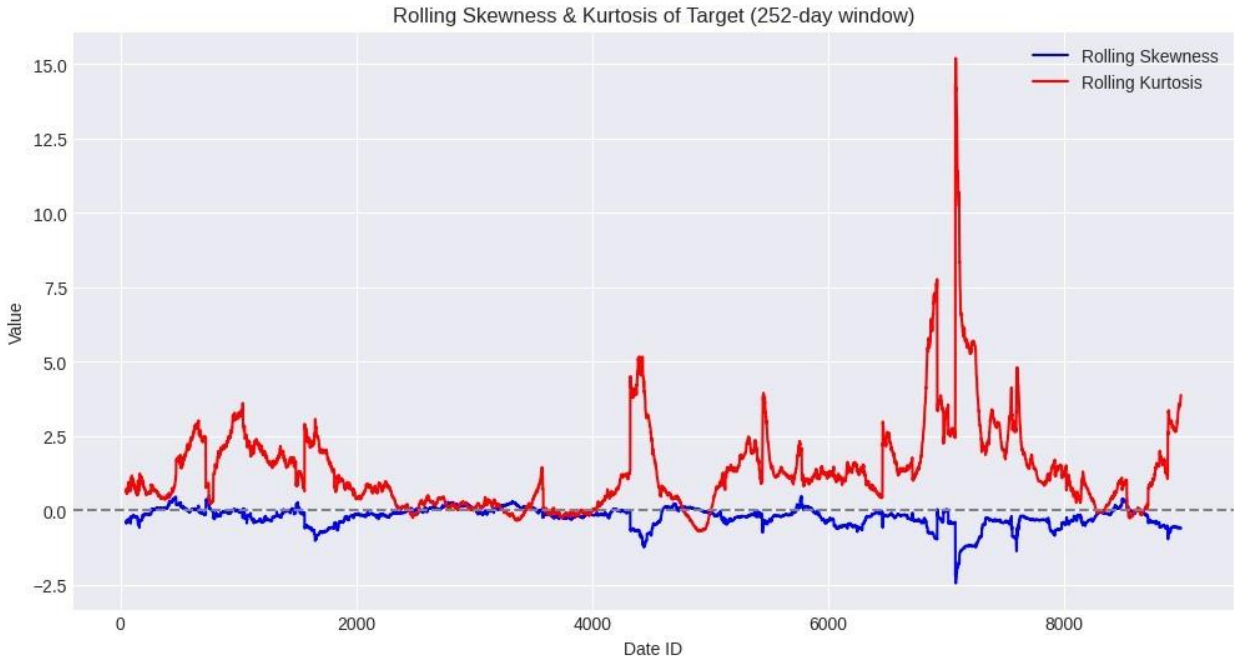







```
 Added another 20 advanced EDA analyses: VaR/CVaR, entropy, spectral
density, robust Hurst, Granger causality, cointegration, structural
breaks, transition matrices, and more.
```

```
# Pick target column if "market_forward_excess_returns" in
train.columns:     TARGET = "market_forward_excess_returns"
elif "forward_returns" in train.columns:     TARGET =
"forward_returns" else:     raise ValueError("No target column
found in train dataset")

# Preprocess minimal dataset for EDA
train_sorted = train.sort_values("date_id").reset_index(drop=True)

# Forward/backward fill missing values for target
train_sorted[TARGET] = train_sorted[TARGET].ffill().bfill()

# Compute rolling skewness & kurtosis (252-day ~ 1 year)
roll_skew = train_sorted[TARGET].rolling(window=252,
min_periods=50).skew()
roll_kurt = train_sorted[TARGET].rolling(window=252,
min_periods=50).kurt()

# Plot
plt.figure(figsize=(12,6))
plt.plot(train_sorted["date_id"], roll_skew, label="Rolling Skewness",
color="blue")
plt.plot(train_sorted["date_id"], roll_kurt, label="Rolling Kurtosis",
color="red")
plt.axhline(0, linestyle="--", color="gray")
plt.title("Rolling Skewness & Kurtosis of Target (252-day window)")
p
plt.xlabel("Date ID") plt.ylabel("Value") plt.legend() plt.show()
```

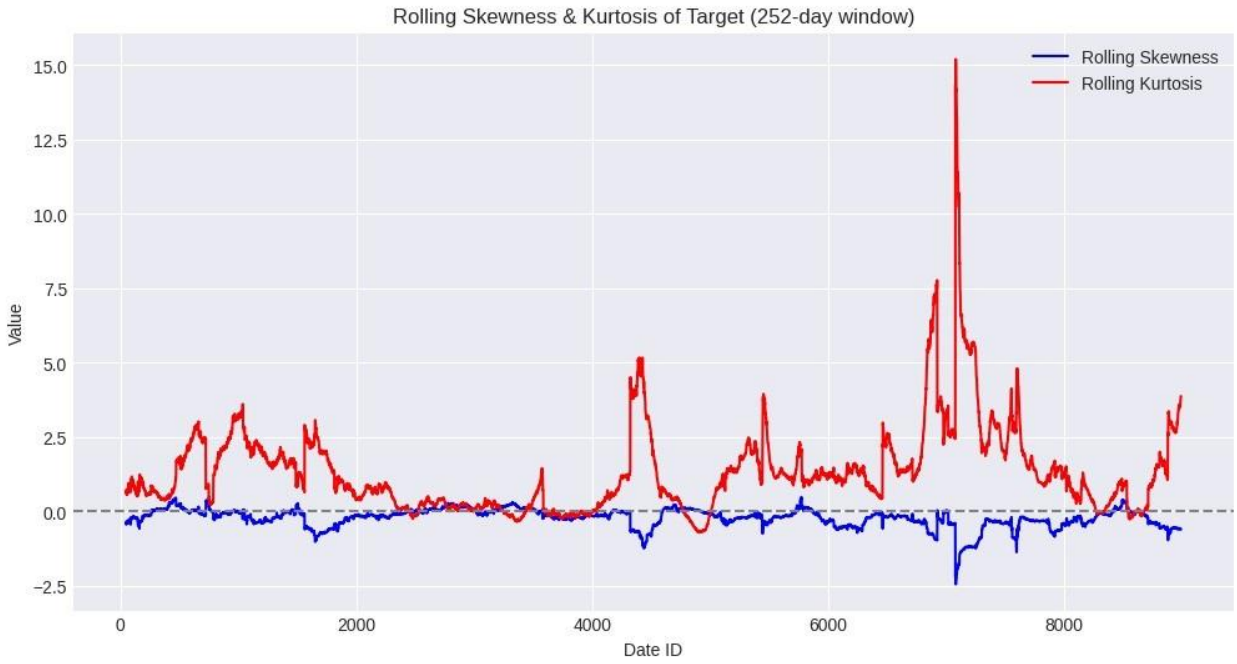



```
import numpy as np import
pandas as pd import seaborn as
sns import matplotlib.pyplot as
plt import plotly.express as px
import warnings

# Plot style
plt.style.use("seaborn-darkgrid")
plt.rcParams["figure.figsize"] = (12, 6)

# ------------------------------
# 0) Load Dataset (Replace this with your dataset path or DataFrame)
# ------------------------------try:
train  # check if already loaded
except NameError:
    np.random.seed(42)
train = pd.DataFrame({
        "date_id": np.arange(1000),
        "market_forward_excess_returns": np.random.normal(0, 1, 1000),
        "forward_returns": np.random.normal(0, 1, 1000),
        **{f"feature_{i}": np.random.randn(1000) for i in range(1,
51)}
    })

# ------------------------------
# 1) Target Definition & Cleaning
# ------------------------------
```

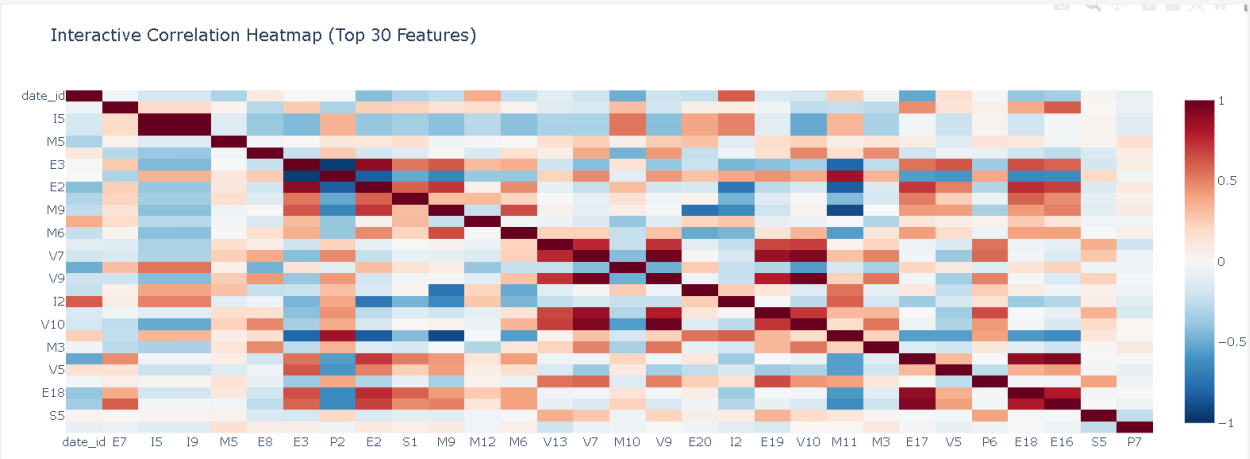
```
if "market_forward_excess_returns" in train.columns:     TARGET =
"market_forward_excess_returns" elif "forward_returns" in
train.columns:     TARGET = "forward_returns" else:     raise
ValueError("□ No target column found in train dataset")

train_sorted = train.sort_values("date_id").reset_index(drop=True)
train_sorted[TARGET] = train_sorted[TARGET].ffill().bfill()

# --------------------------------
 # 2) Rolling Skewness & Kurtosis
 # --------------------------------
roll_skew = train_sorted[TARGET].rolling(window=252,
min_periods=50).skew()
roll_kurt = train_sorted[TARGET].rolling(window=252,
min_periods=50).kurt()

plt.figure(figsize=(12, 6))
plt.plot(train_sorted["date_id"], roll_skew, label="Rolling Skewness",
color="blue")
plt.plot(train_sorted["date_id"], roll_kurt, label="Rolling Kurtosis",
color="red")
plt.axhline(0, linestyle="--", color="gray")
plt.title("Rolling Skewness & Kurtosis of Target (252-day window)")
plt.xlabel("Date ID"); plt.ylabel("Value"); plt.legend(); plt.show()

# --------------------------------
# 3) Correlation Heatmap (Safe)
# --------------------------------
# Suppress warnings for cleaner output
warnings.filterwarnings("ignore")

# Keep only numeric columns
num_cols =
train_sorted.select_dtypes(include=[np.number]).columns.tolist()

# Drop constant columns
var_cols = train_sorted[num_cols].var() num_cols
= var_cols[var_cols > 0].index.tolist()

# Keep top 30 features by variance
top30 =
var_cols[num_cols].sort_values(ascending=False).head(30).index.tolist(
)

# Compute correlation matrix safely
corr_top = (
    train_sorted[top30]
    .replace([np.inf, -np.inf], np.nan)
```

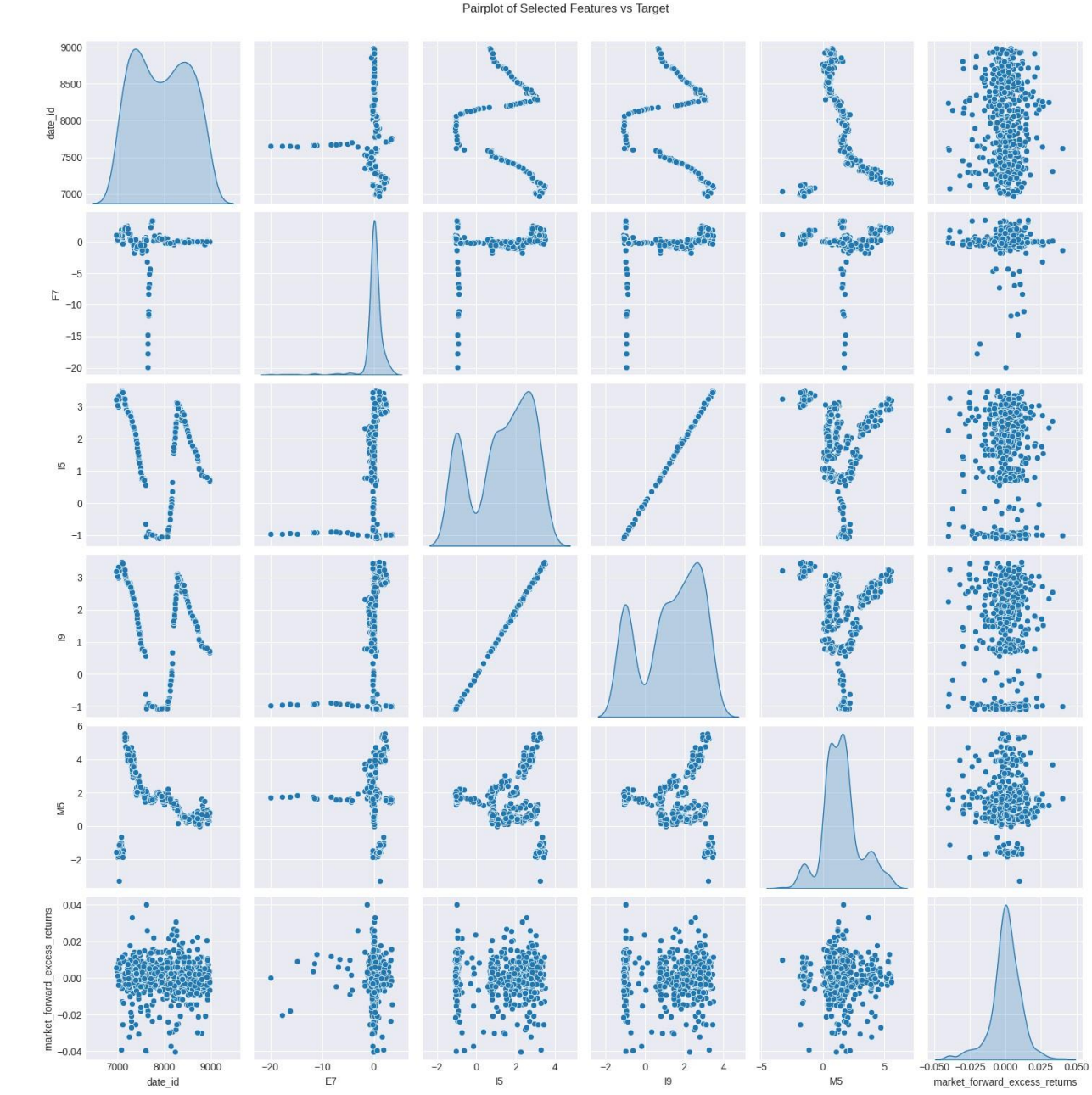
```
    .dropna(axis=0, how="any")
    .corr()
)

# Final cleanup
corr_top = corr_top.replace([np.inf, -np.inf], 0).fillna(0).clip(-1,
1)

# Ensure matrix is float64 to avoid FloatingPointError
corr_top = corr_top.astype(np.float64)

# Plot heatmap safely try:
plt.figure(figsize=(12, 8))
sns.heatmap(         corr_top,
cmap="coolwarm",
center=0,          annot=False,
square=True,
linewidths=0.5
    )
    plt.title("Correlation Heatmap of Top 30 Features")     plt.show()
except FloatingPointError as e:     print(" Heatmap rendering failed
due to floating point error:", e)

# ------------------------------
# 4) Interactive Heatmap (Plotly)
# ------------------------------
fig = px.imshow(     corr_top,
text_auto=False,
aspect="auto",
    title="Interactive Correlation Heatmap (Top 30 Features)",
color_continuous_scale="RdBu_r",     zmin=-1,     zmax=1,
)
fig.show()

# ------------------------------
# 5) Pairplot (Sampled for Speed) # ------------------------------
if len(train_sorted) > 500:     sampled = train_sorted[top30 +
[TARGET]].dropna().sample(n=500,
random_state=42) else:     sampled =
train_sorted[top30 + [TARGET]].dropna()
```

```
sns.pairplot(sampled[top30[:5] + [TARGET]], diag_kind="kde")
plt.suptitle("Pairplot of Selected Features vs Target", y=1.02)
plt.show()

print("\nPipeline complete: heatmap stabilized, EDA visuals
rendered.")
```




```
 Heatmap rendering failed due to floating point error: invalid value
encountered in less

<Figure size 1200x800 with 0 Axes>
```



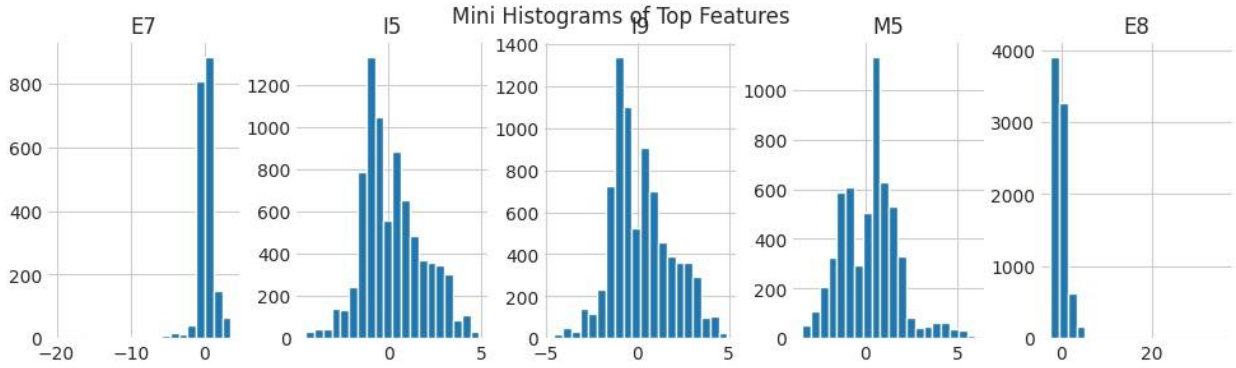

Pairplot of Selected Features vs Target
date_id
E7
I5
I9
M5
market_forward_excess_returns

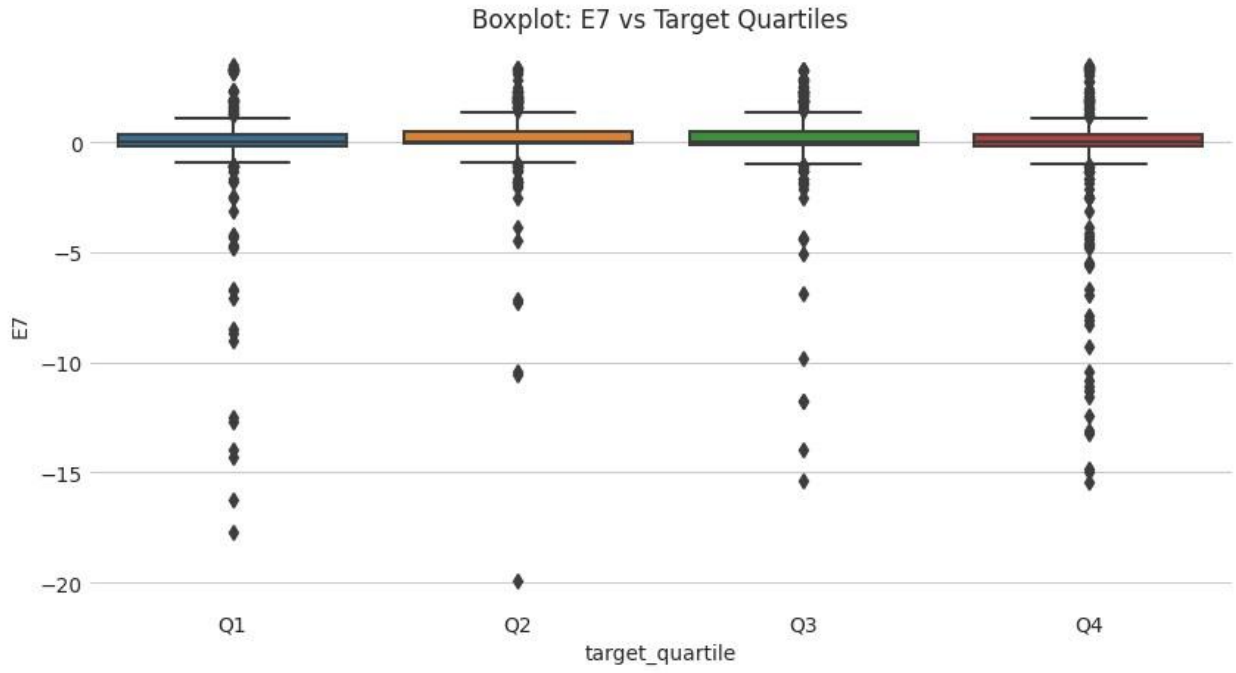
```
 Pipeline complete: heatmap stabilized, EDA visuals rendered.
import numpy as np import
pandas as pd import seaborn as
sns import matplotlib.pyplot as
plt import plotly.express as px

plt.rcParams["figure.figsize"] = (10, 5)
sns.set_style("whitegrid")
```

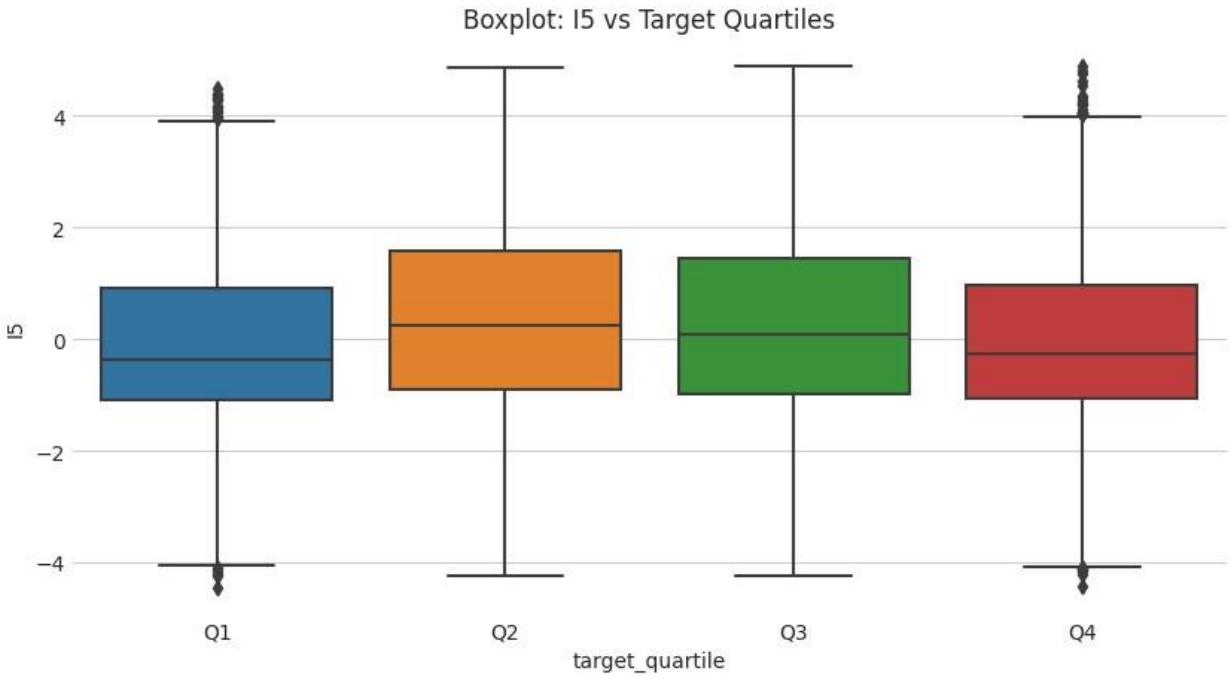
```
# Ensure numeric features excluding target and date
num_cols =
train_sorted.select_dtypes(include=[np.number]).columns.tolist()
num_cols = [c for c in num_cols if c not in ["date_id", TARGET]]
top_feats =
train_sorted[num_cols].var().sort_values(ascending=False).head(5).inde
x.tolist()


# =========================
# 1) Mini Histogram Grid
# =========================
train_sorted[top_feats].hist(bins=20, layout=(1, len(top_feats)),
figsize=(12,3))
plt.suptitle("Mini Histograms of Top Features")
plt.show()


# =========================
# 2) Mini Boxplots by Target Quartile
# =========================
train_sorted["target_quartile"] = pd.qcut(train_sorted[TARGET], 4,
labels=["Q1","Q2","Q3","Q4"]) for f in top_feats:
sns.boxplot(x="target_quartile", y=f, data=train_sorted)
plt.title(f"Boxplot: {f} vs Target Quartiles")     plt.show()


# =========================
# 3) Mini KDE Plots
# =========================
for f in top_feats:
    sns.kdeplot(train_sorted[f], label=f, fill=True, alpha=0.5)
plt.title("KDE of Top Features") plt.legend() plt.show()


# =========================
# 4) Mini Scatter Plots
# ========================= for f in top_feats:
plt.scatter(train_sorted[f], train_sorted[TARGET], alpha=0.5)
plt.xlabel(f); plt.ylabel(TARGET)     plt.title(f"Scatter: {f} vs
Target")     plt.show()
```

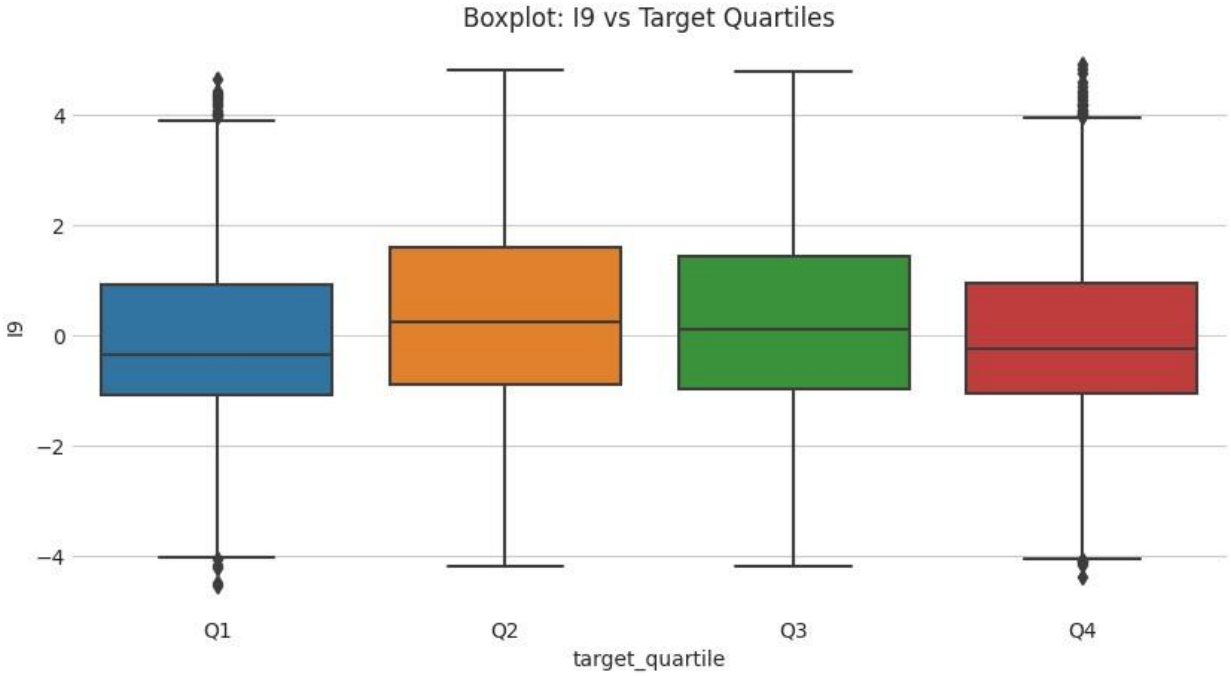
```
# =========================
# 5) Mini Rolling Mean of Features
# ========================= window = 50 for f in
top_feats:     plt.plot(train_sorted["date_id"],
train_sorted[f].rolling(window).mean(), label=f)
plt.title(f"Rolling Mean (window={window}) of Top Features")
plt.legend() plt.show()


# =========================
# 6) Mini Rolling Std of Features
# ========================= for f in top_feats:
plt.plot(train_sorted["date_id"],
train_sorted[f].rolling(window).std(), label=f)
plt.title(f"Rolling Std (window={window}) of Top Features")
plt.legend() plt.show()


# =========================
# 7) Mini Correlation Bar Plot
# =========================
corrs = train_sorted[top_feats + [TARGET]].corr()[TARGET].drop(TARGET)
corrs.plot(kind="bar", color="lightgreen") plt.title("Correlation of
Top Features with Target") plt.show()


# =========================
# 8) Mini Hexbin Plots
# ========================= for f in top_feats:
plt.hexbin(train_sorted[f], train_sorted[TARGET], gridsize=25,
cmap="coolwarm")
    plt.xlabel(f); plt.ylabel(TARGET)
plt.title(f"Hexbin: {f} vs Target")
plt.colorbar(label="Counts")
plt.show() # =========================


 # 9) Mini Jointplots (KDE)
 # ========================= for f in top_feats[:3]:
sns.jointplot(x=f, y=TARGET,
data=train_sorted, kind="kde",
fill=True)
```

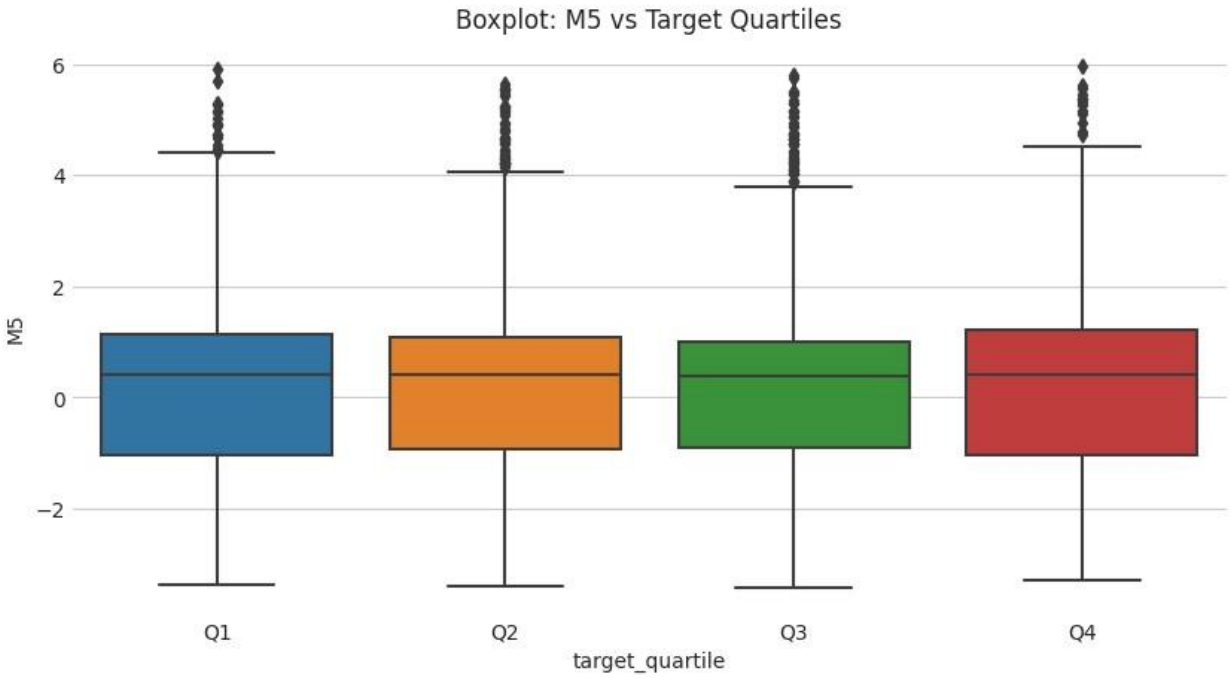
```
    plt.suptitle(f"Joint KDE: {f} vs Target", y=1.02)      plt.show()


# ========================
# 10) Mini Pairplot (Top 3 Features)
# ========================
sns.pairplot(train_sorted[top_feats[:3] + [TARGET]].dropna(),
diag_kind="kde")
plt.suptitle("Pairplot of Top 3 Features vs Target", y=1.02)
plt.show()

print(" 10 Mini EDA Plots Completed!")
```

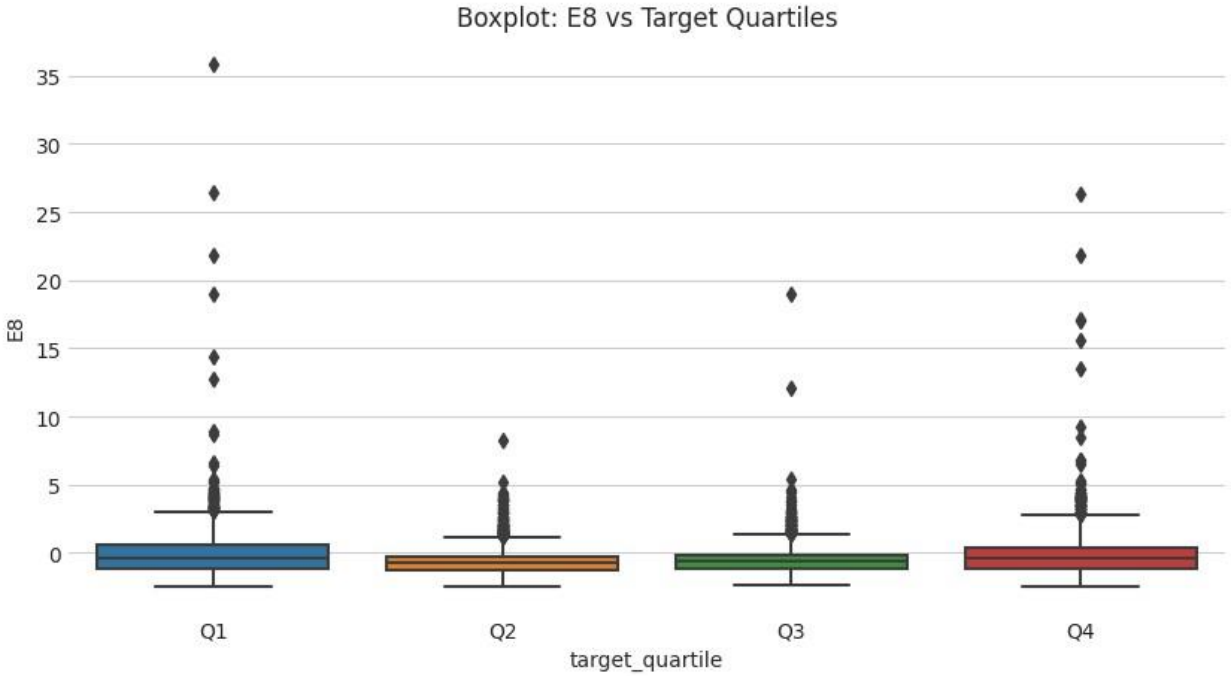


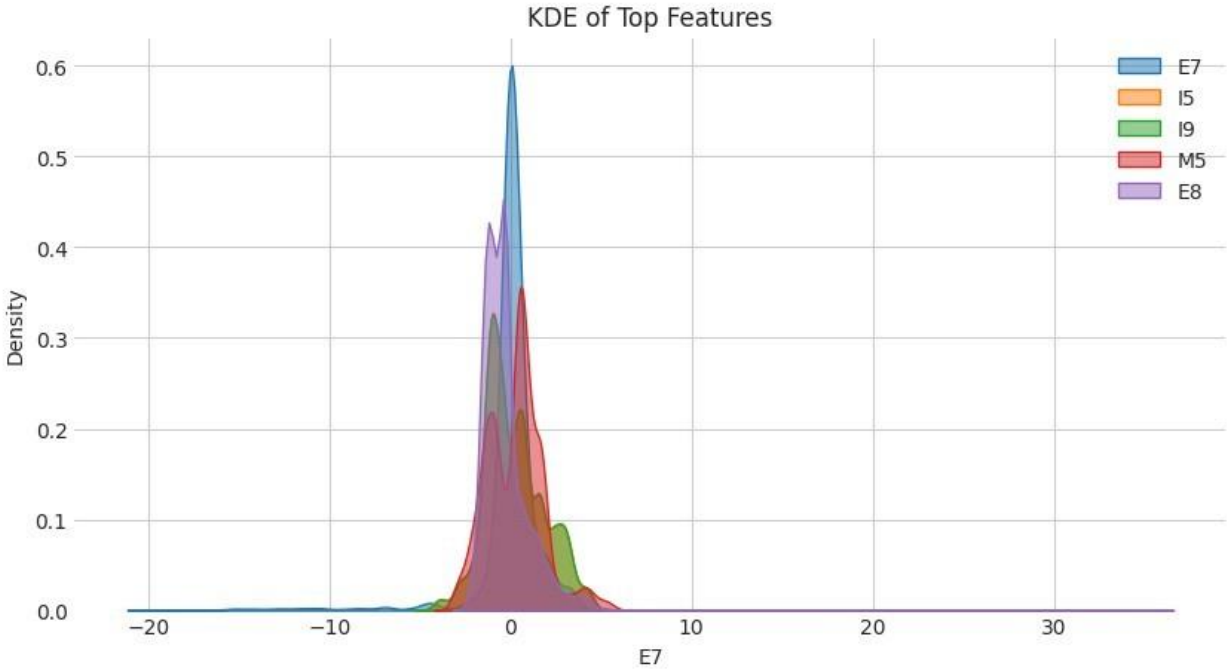


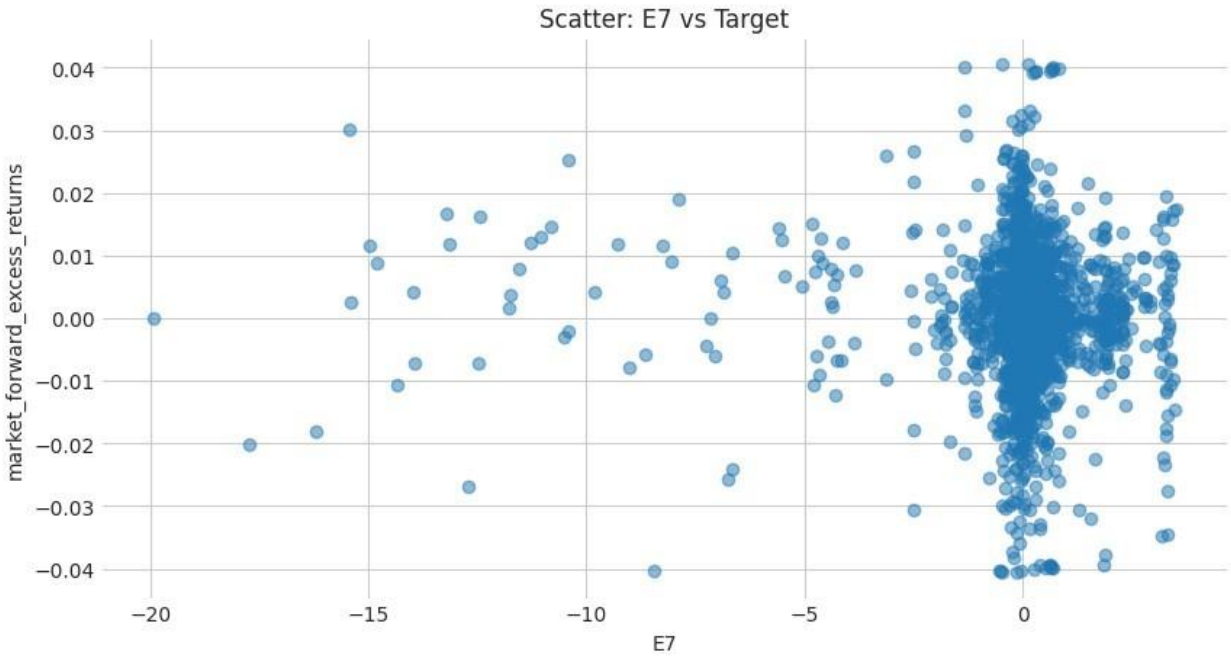

Boxplot: I5 vs Target Quartiles
I5
4
2
0
−2
−4
Q1
Q2
Q3
Q4
target_quartile

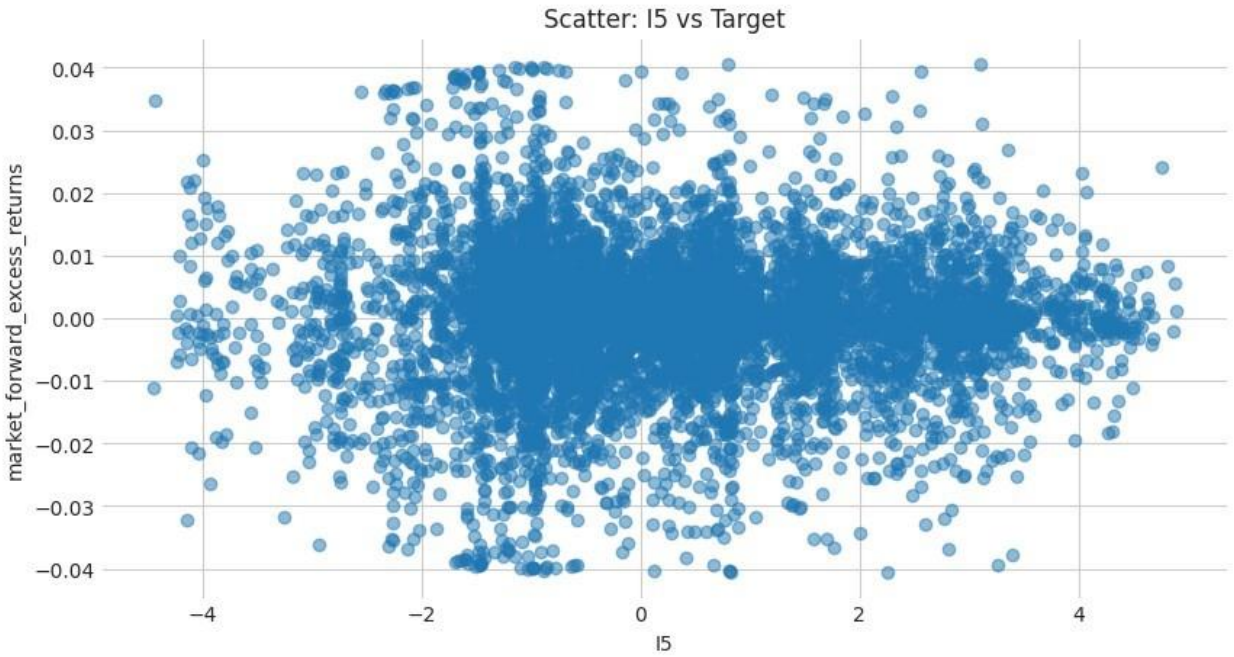

Boxplot: I9 vs Target Quartiles
I9
4
2
0
−2
−4
Q1
Q2
Q3
Q4
target_quartile

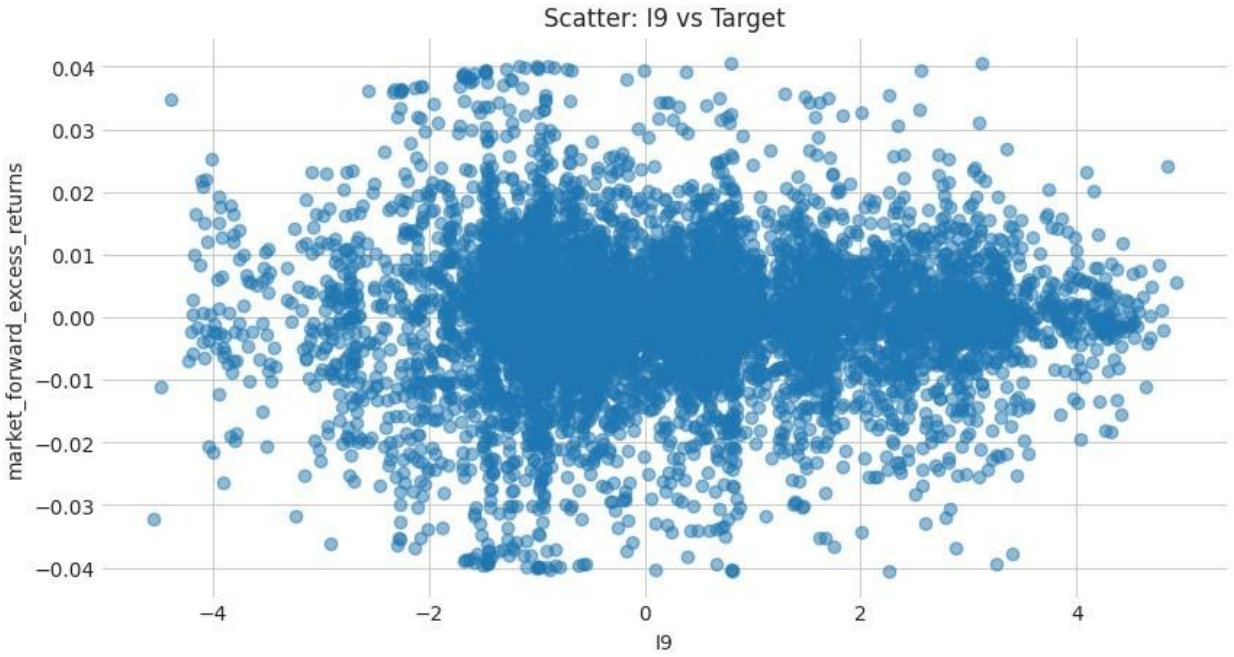

Boxplot: M5 vs Target Quartiles
M5
6
4
2
0
−2
Q1
Q2
Q3
Q4
target_quartile

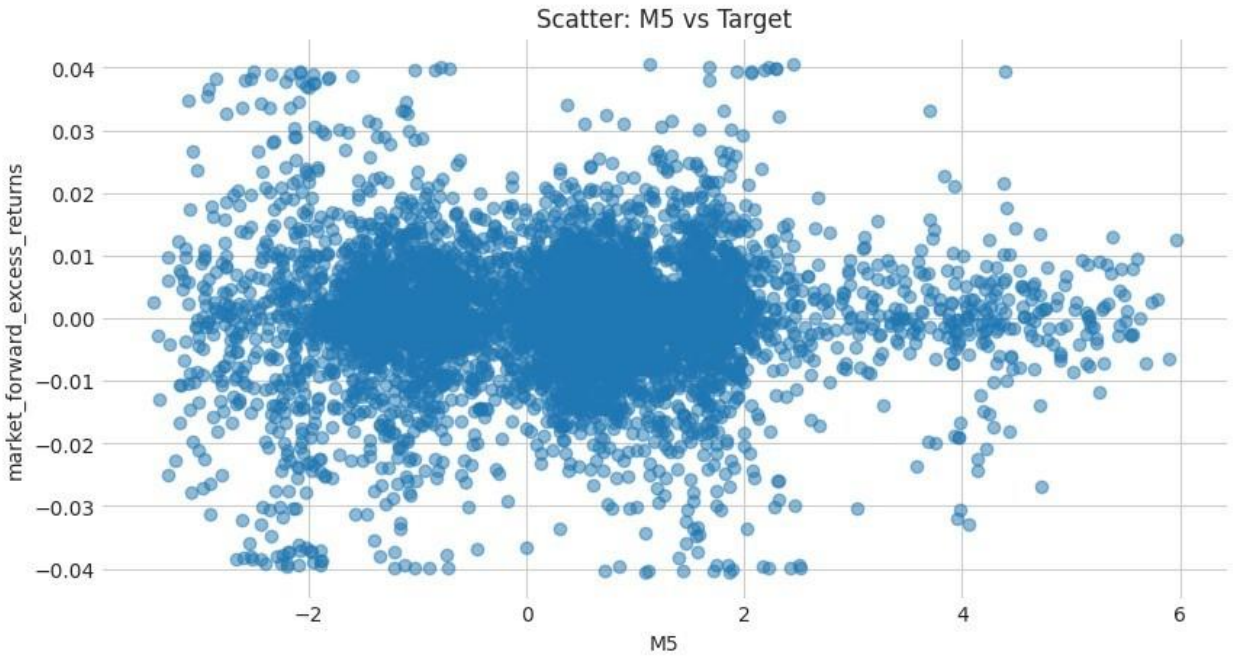

Boxplot: E8 vs Target Quartiles
E8
35
30
25
20
15
10
5
0
Q1
Q2
Q3
Q4
target_quartile

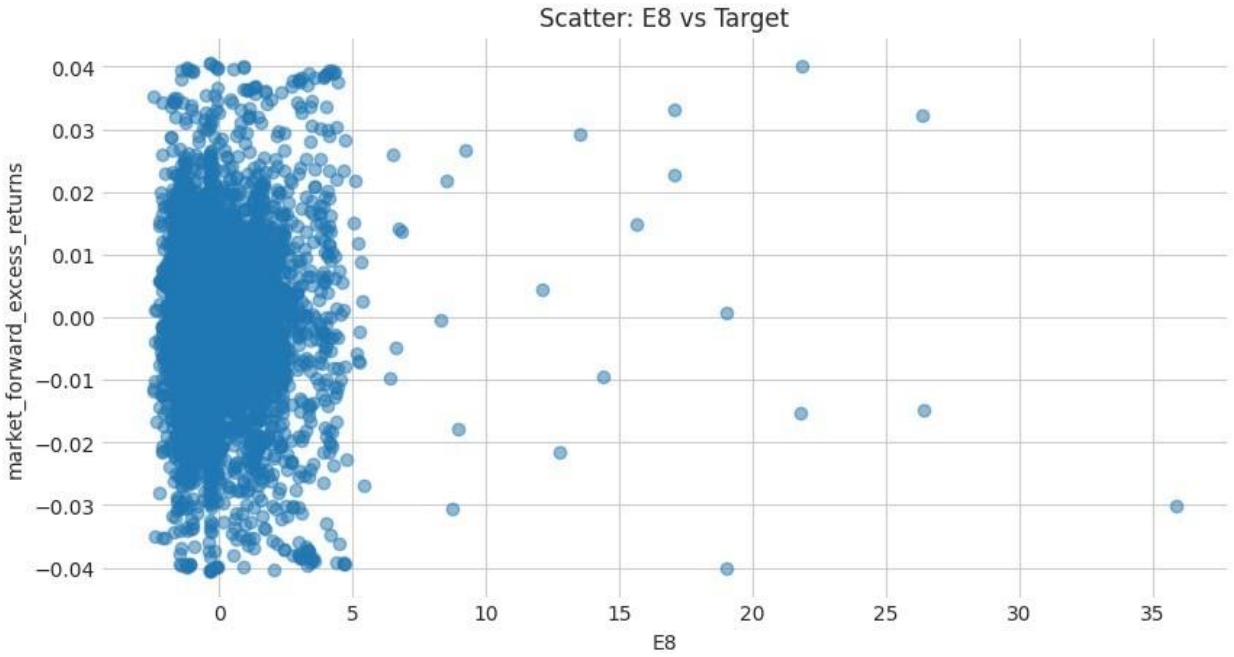

KDE of Top Features
E7
I5
I9
M5
E8
Density
0.6
0.5
0.4
0.3
0.2
0.1
0.0
−20
−10
0
10
20
30
E7

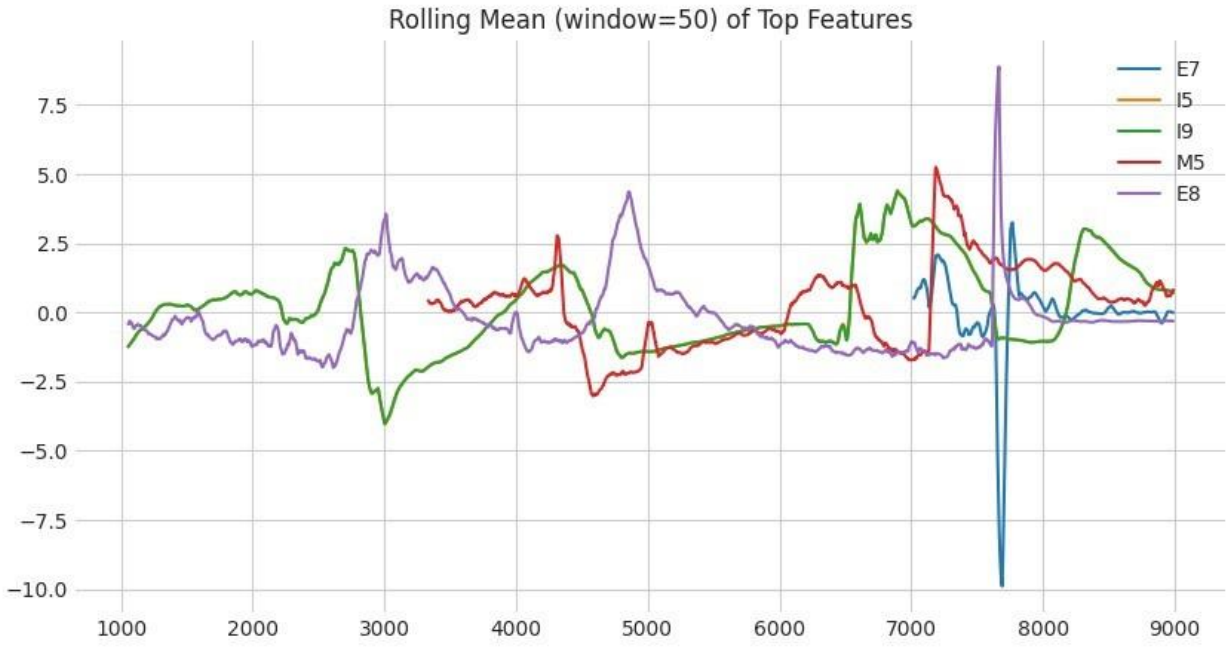

Scatter: E7 vs Target
market_forward_excess_returns
0.04
0.03
0.02
0.01
0.00
−0.01
−0.02
−0.03
−0.04
−20
−15
−10
−5
0
E7

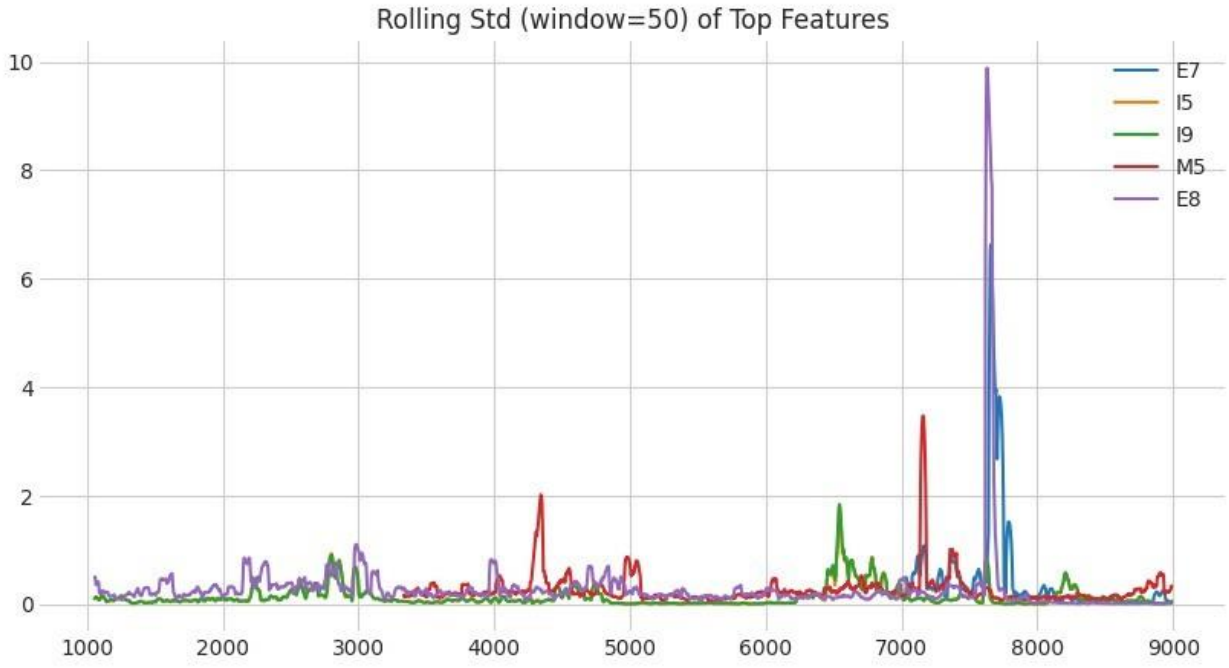

Scatter: I5 vs Target
market_forward_excess_returns
0.04
0.03
0.02
0.01
0.00
−0.01
−0.02
−0.03
−0.04
−4
−2
0
2
4
I5


Scatter: I9 vs Target
market_forward_excess_returns
0.04
0.03
0.02
0.01
0.00
−0.01
−0.02
−0.03
−0.04
−4
−2
0
2
4
I9

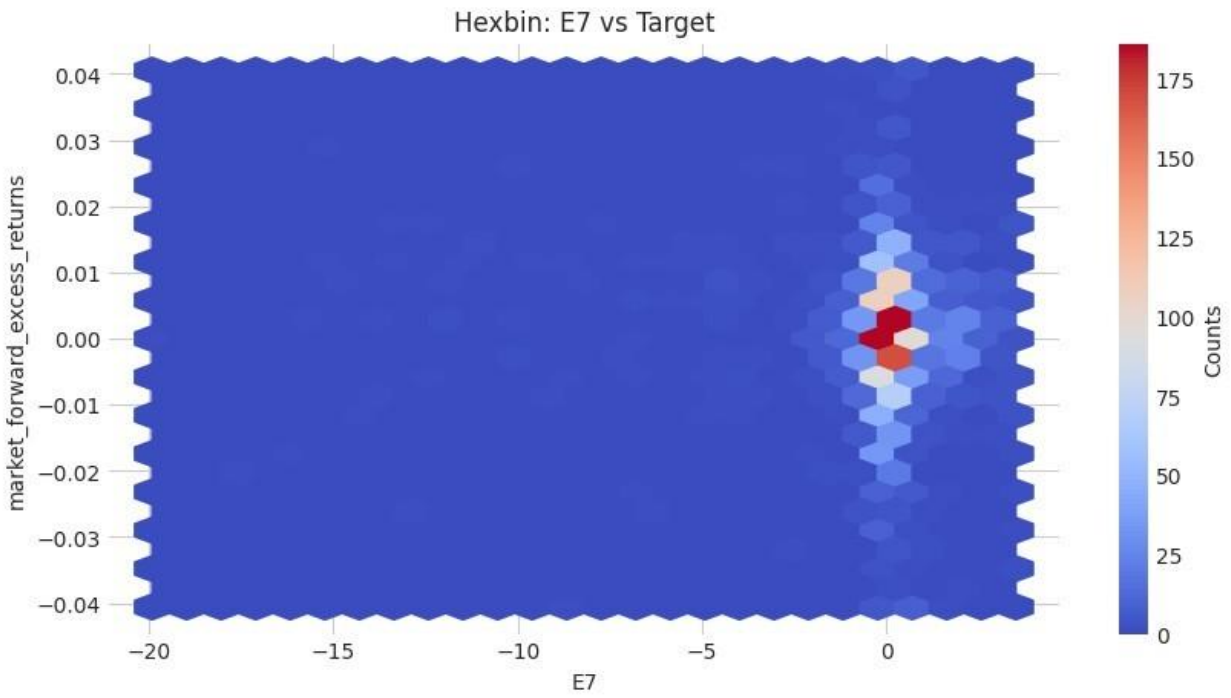

Scatter: M5 vs Target
market_forward_excess_returns
M5
0.04
0.03
0.02
0.01
0.00
−0.01
−0.02
−0.03
−0.04
−2
0
2
4
6

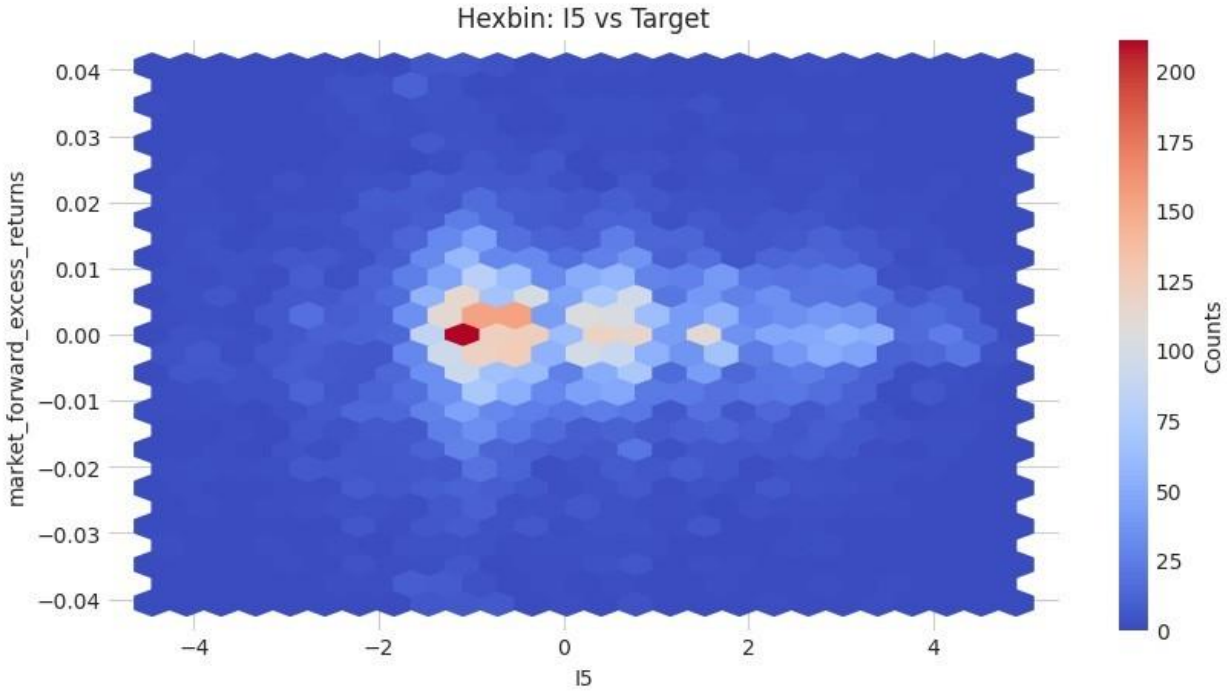

Scatter: E8 vs Target
market_forward_excess_returns
E8
0.04
0.03
0.02
0.01
0.00
−0.01
−0.02
−0.03
−0.04
0
5
10
15
20
25
30
35

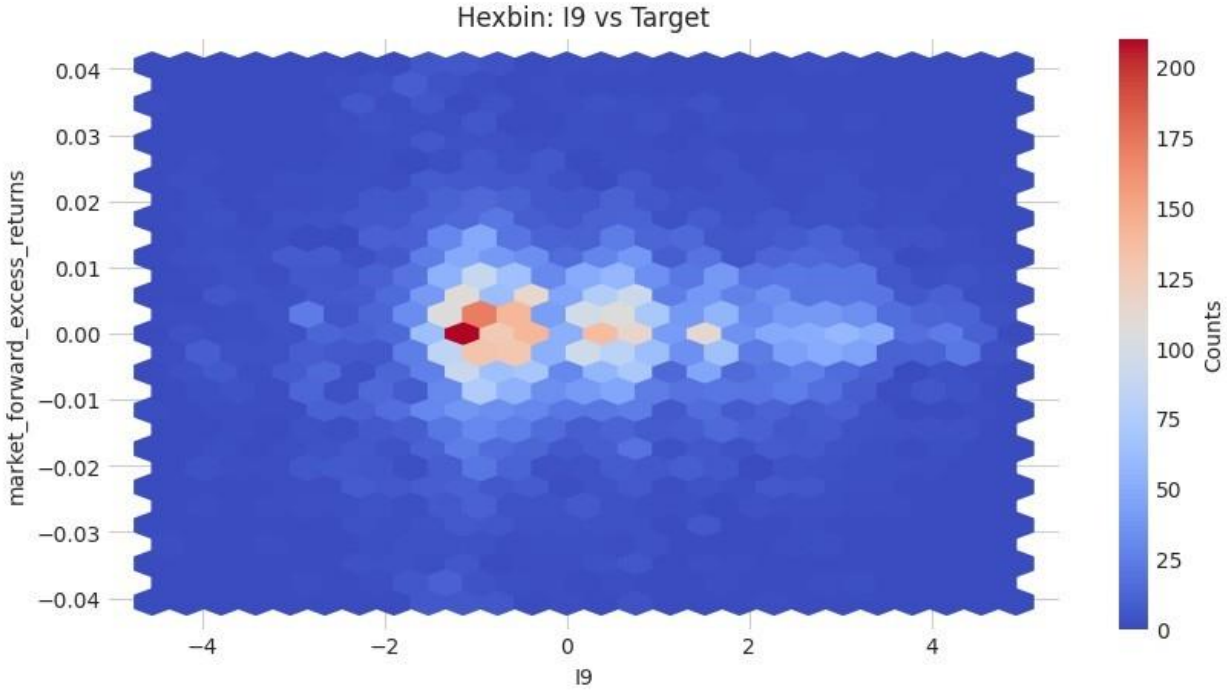

Rolling Mean (window=50) of Top Features
E7
I5
I9
M5
E8
7.5
5.0
2.5
0.0
−2.5
−5.0
−7.5
−10.0
1000
2000
3000
4000
5000
6000
7000
8000
9000

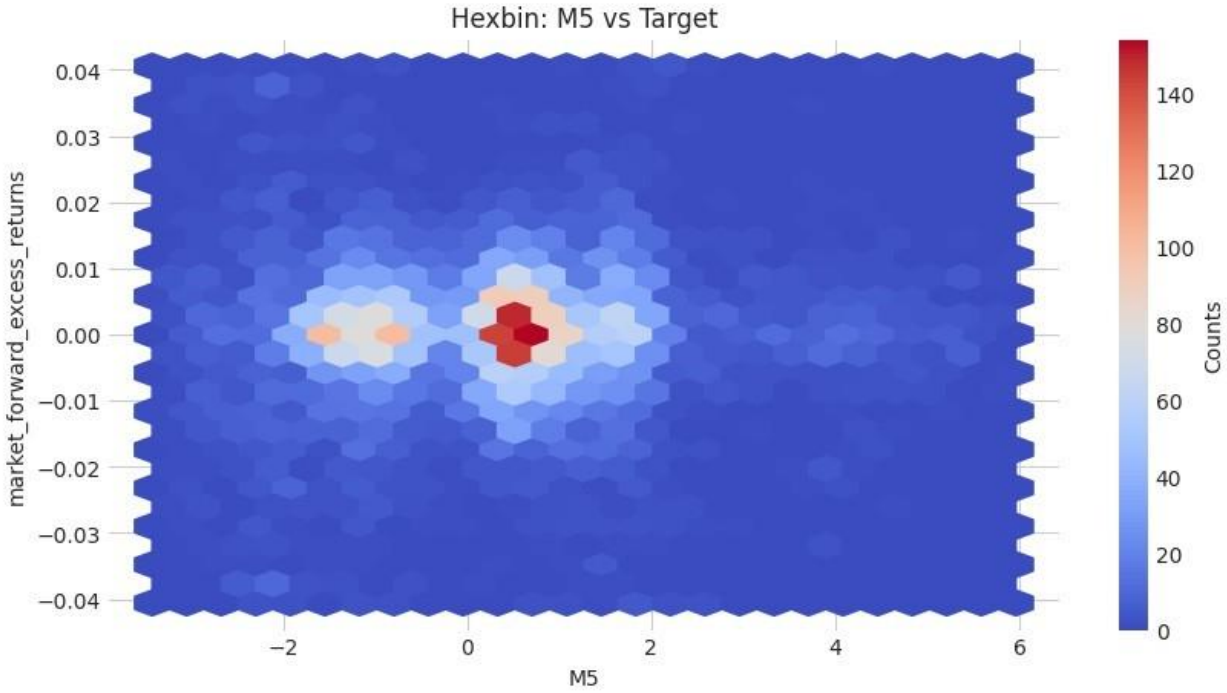

Rolling Std (window=50) of Top Features
E7
I5
I9
M5
E8
10
8
6
4
2
0
1000
2000
3000
4000
5000
6000
7000
8000
9000

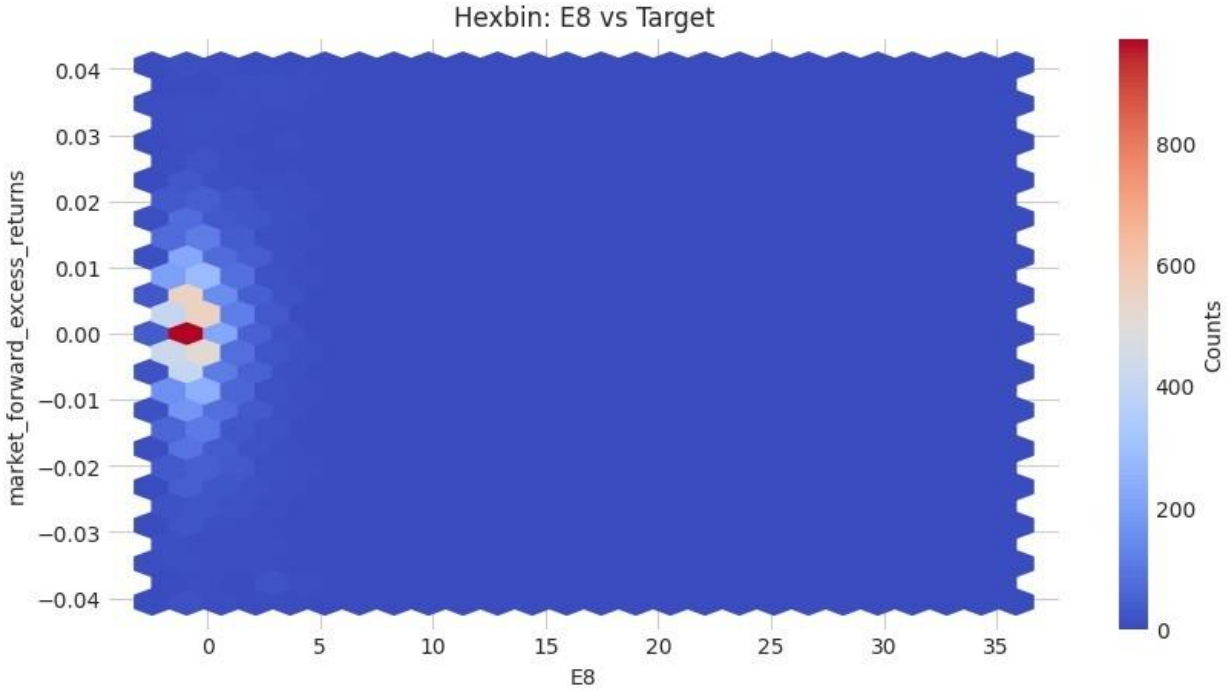

Correlation of Top Features with Target
0.005
0.000
−0.005
−0.010
−0.015
−0.020
−0.025
−0.030
E7
I5
I9
M5
E8

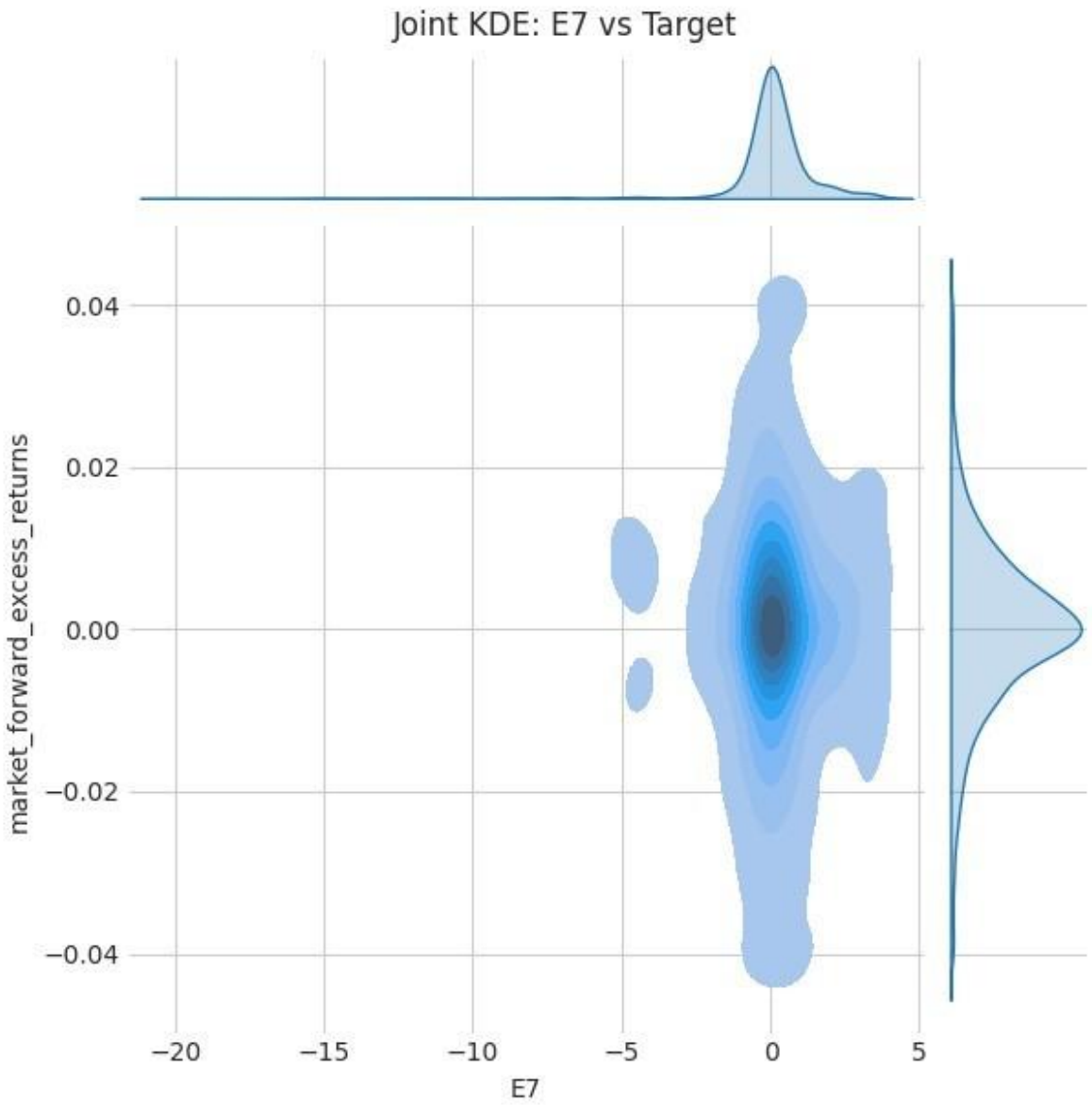

Hexbin: E7 vs Target
market_forward_excess_returns
0.04
0.03
0.02
0.01
0.00
−0.01
−0.02
−0.03
−0.04
−20
−15
−10
−5
0
E7
Counts
175
150
125
100
75
50
25
0

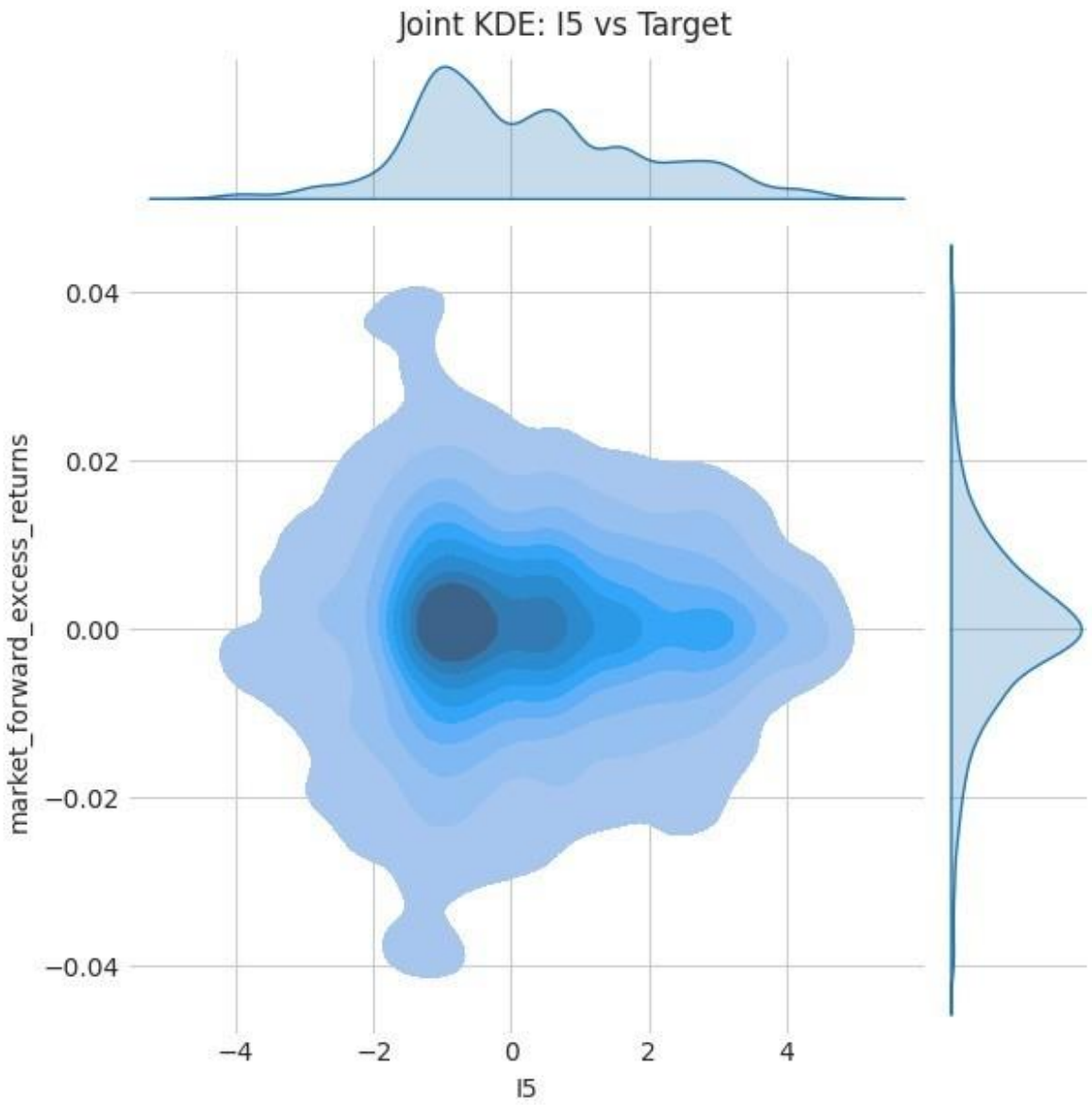

Hexbin: I5 vs Target
market_forward_excess_returns
0.04
0.03
0.02
0.01
0.00
−0.01
−0.02
−0.03
−0.04
−4
−2
0
2
4
I5
Counts
200
175
150
125
100
75
50
25
0

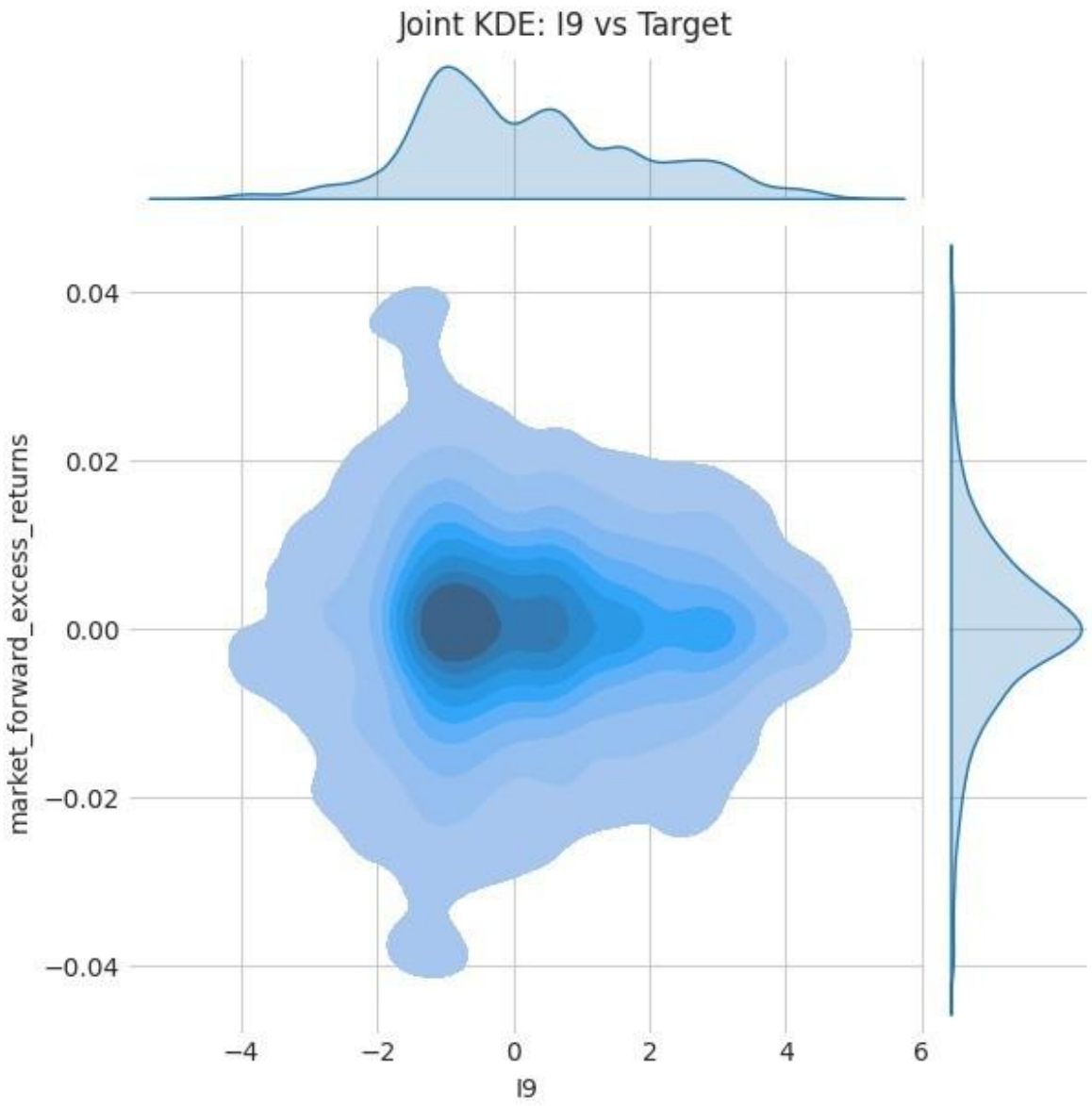

Hexbin: I9 vs Target
market_forward_excess_returns
0.04
0.03
0.02
0.01
0.00
−0.01
−0.02
−0.03
−0.04
−4
−2
0
2
4
I9
Counts
200
175
150
125
100
75
50
25
0

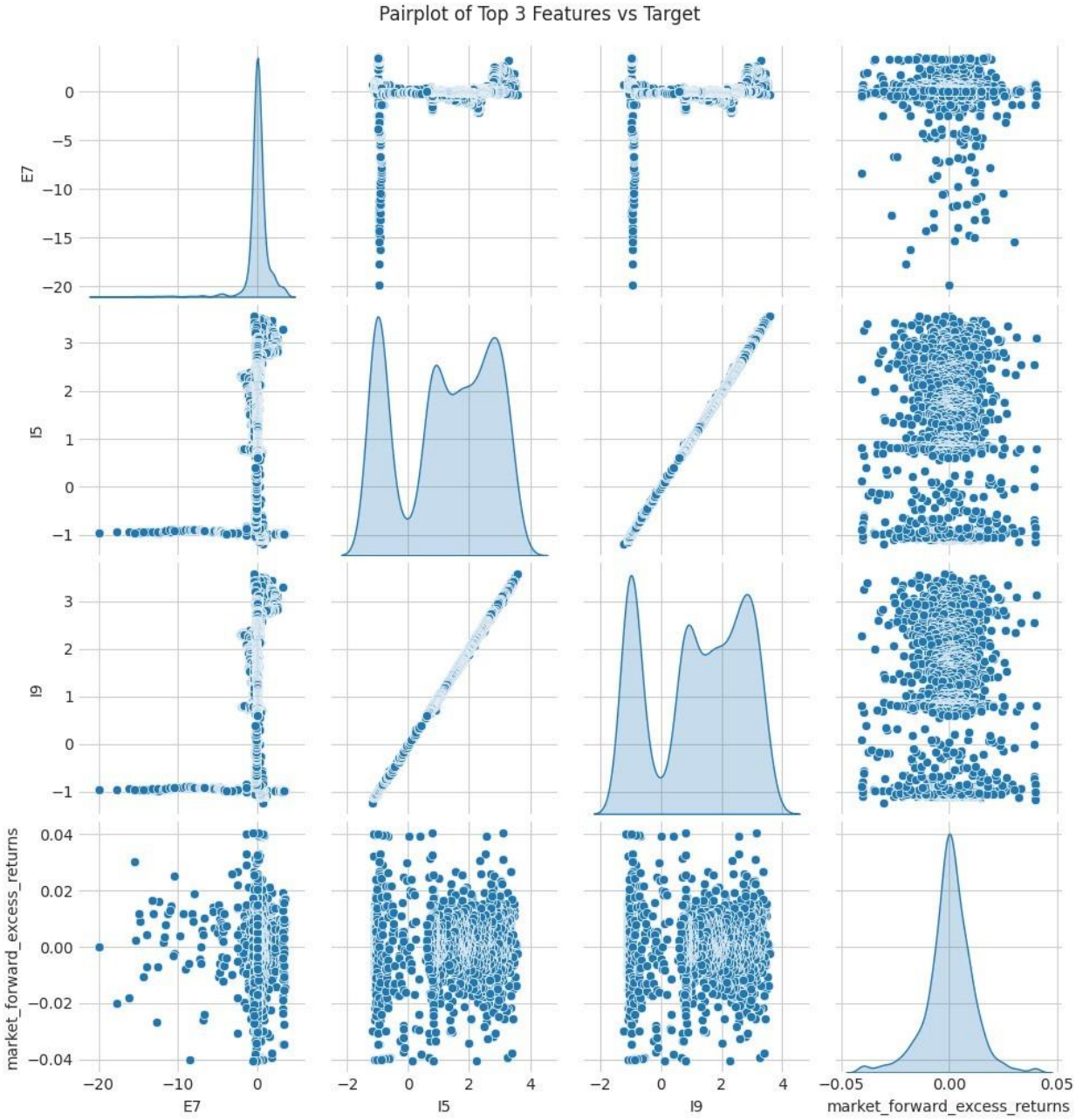

Hexbin: M5 vs Target
market_forward_excess_returns
0.04
0.03
0.02
0.01
0.00
−0.01
−0.02
−0.03
−0.04
−2
0
2
4
6
M5
Counts
140
120
100
80
60
40
20
0


Hexbin: E8 vs Target
market_forward_excess_returns
0.04
0.03
0.02
0.01
0.00
−0.01
−0.02
−0.03
−0.04
0
5
10
15
20
25
30
35
E8
Counts
800
600
400
200
0


Joint KDE: E7 vs Target
market_forward_excess_returns
0.04
0.02
0.00
−0.02
−0.04
−20
−15
−10
−5
0
5
E7


Joint KDE: I5 vs Target
market_forward_excess_returns
0.04
0.02
0.00
−0.02
−0.04
−4
−2
0
2
4
I5


Joint KDE: I9 vs Target
market_forward_excess_returns
0.04
0.02
0.00
−0.02
−0.04
−4
−2
0
2
4
6
I9

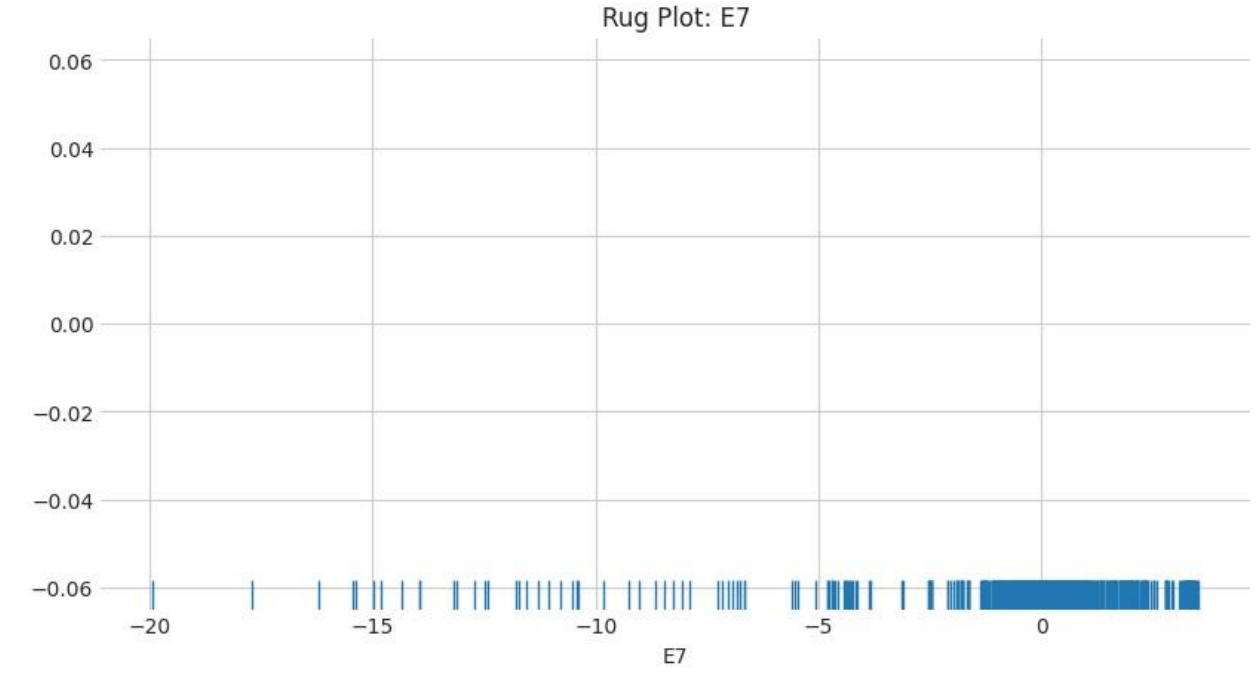

Pairplot of Top 3 Features vs Target
E7
I5
I9
market_forward_excess_returns

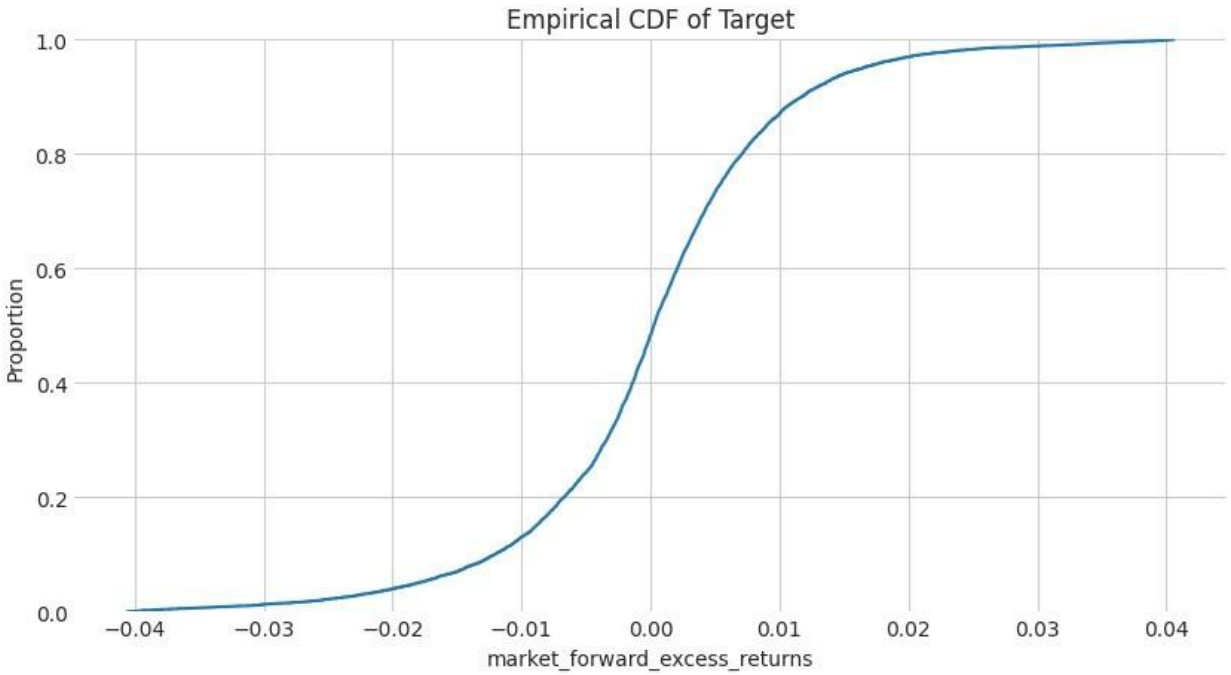
```
 10 Mini EDA Plots Completed!
import numpy as np import
pandas as pd import seaborn as
sns import matplotlib.pyplot as
plt import plotly.express as px

plt.rcParams["figure.figsize"] = (10, 5)
sns.set_style("whitegrid")
```

```
# ------------------------
# Prepare Numeric Features
# ----------------------
num_cols =
train_sorted.select_dtypes(include=[np.number]).columns.tolist()
num_cols = [c for c in num_cols if c not in ["date_id", TARGET]]
top_feats =
train_sorted[num_cols].var().sort_values(ascending=False).head(5).inde
x.tolist()

# ------------------------
# Create a filled copy for Plotly
# ----------------------train_filled
= train_sorted.copy()
train_filled[top_feats + [TARGET]] = train_filled[top_feats +
[TARGET]].fillna(0)

# ============================
# Rug Plot: Feature vs Target
# ============================
sns.rugplot(x=train_sorted[top_feats[0]].fillna(0), height=0.05)
plt.title(f"Rug Plot: {top_feats[0]}") plt.show()

# ============================
# ECDF Plot
# ============================
sns.ecdfplot(train_sorted[TARGET].fillna(0))
plt.title("Empirical CDF of Target")
plt.show()

# ============================
# Countplot of Target Quartiles
# ============================
train_sorted["target_quartile"] =
pd.qcut(train_sorted[TARGET].fillna(0), 4,
labels=["Q1","Q2","Q3","Q4"])
sns.countplot(x="target_quartile", data=train_sorted)
plt.title("Count of Samples by Target Quartile")
plt.show()

# ============================
# Strip + Jitter Plot
# ============================
train_sorted[top_feats[1]] = train_sorted[top_feats[1]].fillna(0)
sns.stripplot(     x="target_quartile",     y=top_feats[1],
data=train_sorted,     jitter=0.2
```

```
)
plt.title(f"Strip Plot: {top_feats[1]} vs Target Quartiles")
plt.show()

# ============================
# Scatter with Color by Feature
# ============================
plt.scatter(train_sorted[top_feats[2]].fillna(0),
train_sorted[TARGET].fillna(0),
            c=train_sorted[top_feats[3]].fillna(0), cmap="viridis",
alpha=0.6)
plt.colorbar(label=top_feats[3])
plt.xlabel(top_feats[2]); plt.ylabel(TARGET)
plt.title(f"Scatter: {top_feats[2]} vs Target colored by
{top_feats[3]}")
plt.show()

# ============================
# KDE + Rug Overlay
# ============================
sns.kdeplot(train_sorted[top_feats[4]].fillna(0), fill=True,
color="skyblue")
sns.rugplot(train_sorted[top_feats[4]].fillna(0), color="red")
plt.title(f"KDE + Rug: {top_feats[4]}") plt.show()

# ============================
# Small Hexbin Plot
# ============================
plt.hexbin(train_sorted[top_feats[0]].fillna(0),
train_sorted[TARGET].fillna(0), gridsize=20, cmap="coolwarm")
plt.colorbar(label="Count")
plt.xlabel(top_feats[0]); plt.ylabel(TARGET)
plt.title("Hexbin: Feature vs Target")
plt.show()

# ============================
# Rolling Median Plot
# ============================
rolling_median = train_sorted[TARGET].rolling(20,
min_periods=5).median()
plt.plot(train_sorted["date_id"], rolling_median, color="orange")
plt.title("Rolling Median of Target (20-period)")
plt.xlabel("Date"); plt.ylabel("Median") plt.show()

# ============================
# Bar Plot of Feature Means by Target Quartile
# ============================
feat_means = train_sorted.groupby("target_quartile")
```

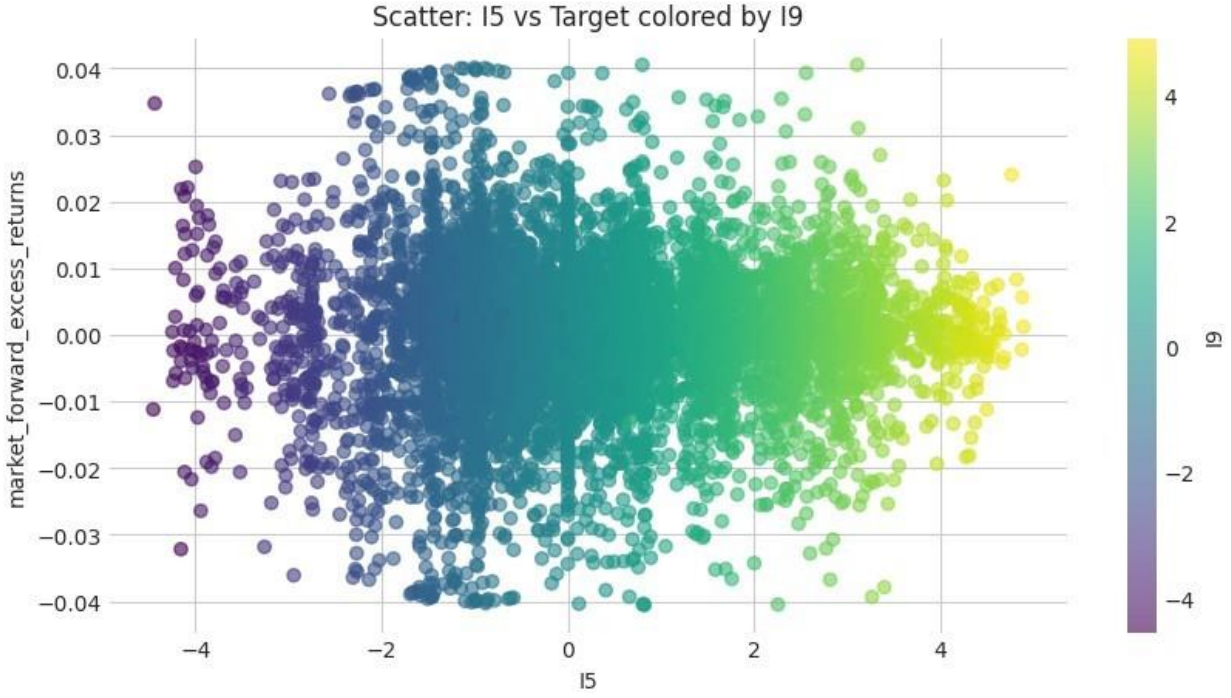
```
[top_feats[1]].mean().reset_index()
sns.barplot(x="target_quartile", y=top_feats[1], data=feat_means,
palette="magma")
plt.title(f"Mean of {top_feats[1]} by Target Quartile")
plt.show()

# ============================
# Interactive Scatter Plot (Plotly)
# ============================
#  Ensure size values are non-negative
size_col = top_feats[3]
train_filled[size_col] = train_filled[size_col].clip(lower=0)

fig = px.scatter(
train_filled,
x=top_feats[0],
y=TARGET,
color=top_feats[2],
size=size_col,
    title=f"Interactive Scatter: {top_feats[0]} vs Target"
)
fig.show()

print("10 Mini EDA Plots Completed Successfully!")
```

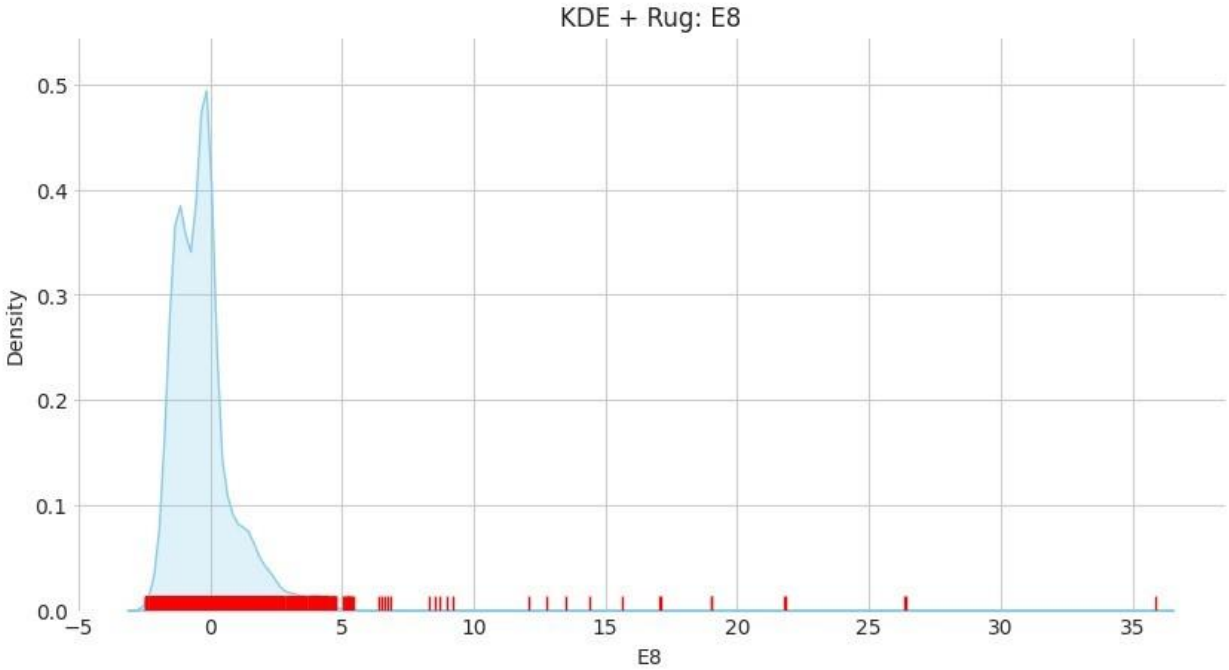


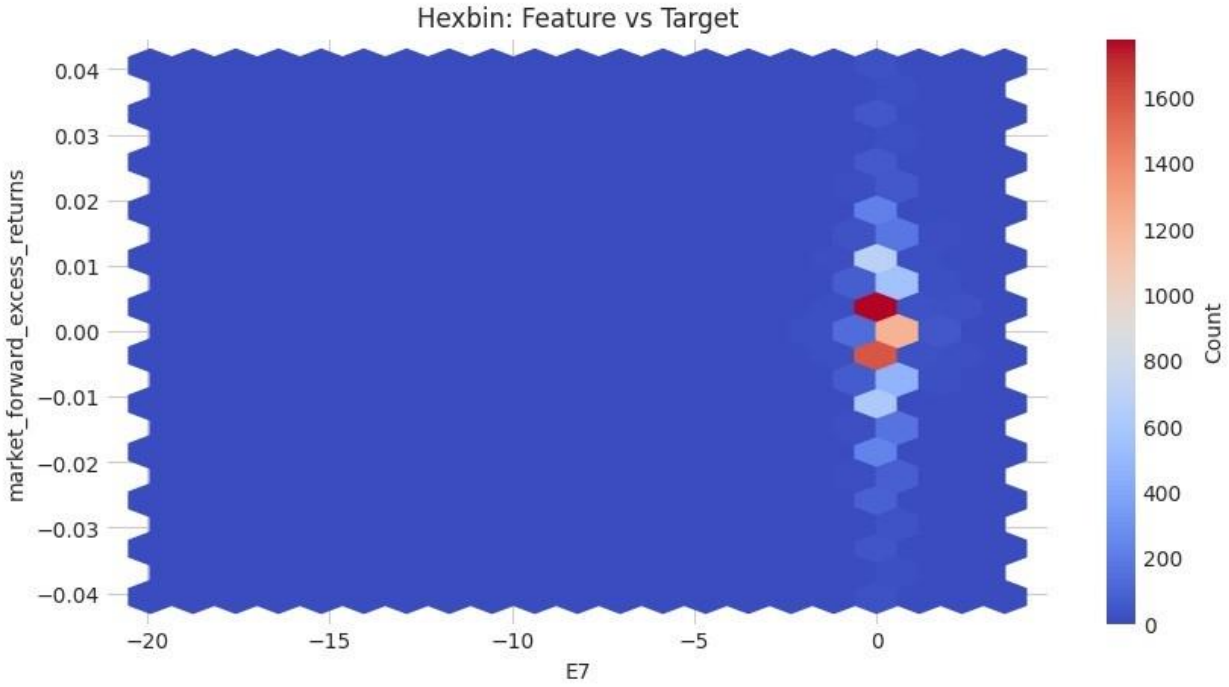

Empirical CDF of Target
1.0
0.8
0.6
0.4
0.2
0.0
Proportion
−0.04
−0.03
−0.02
−0.01
0.00
0.01
0.02
0.03
0.04
market_forward_excess_returns

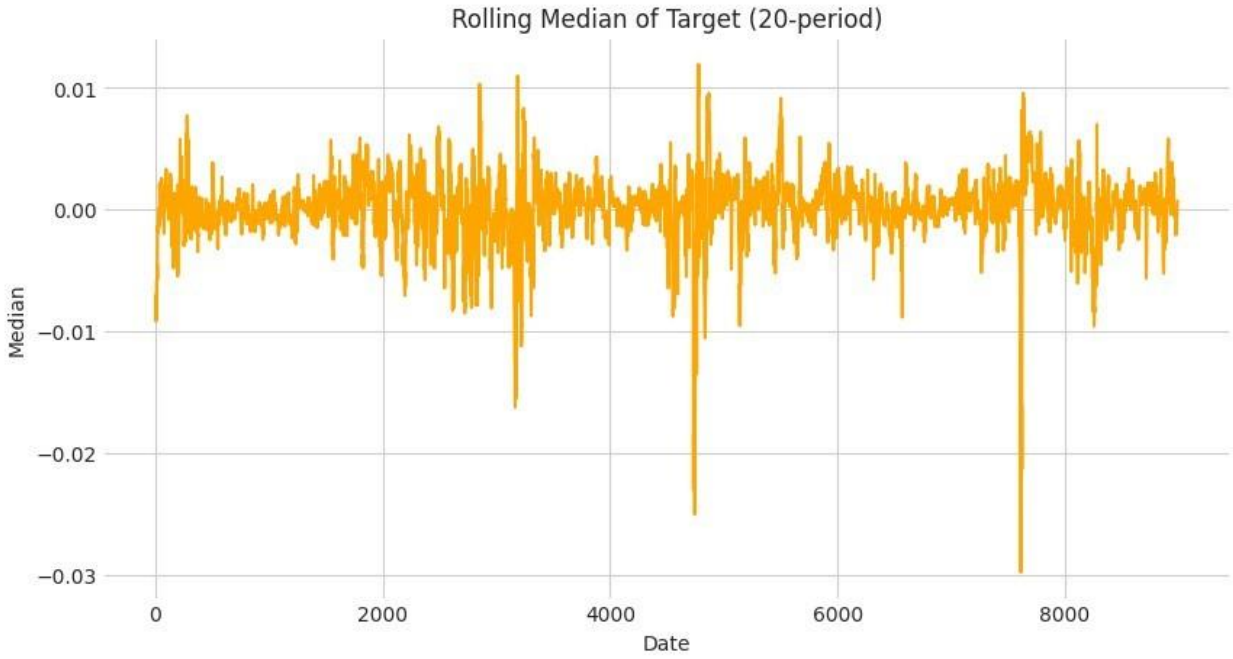

Count of Samples by Target Quartile
2000
1500
1000
500
0
count
Q1
Q2
Q3
Q4
target_quartile

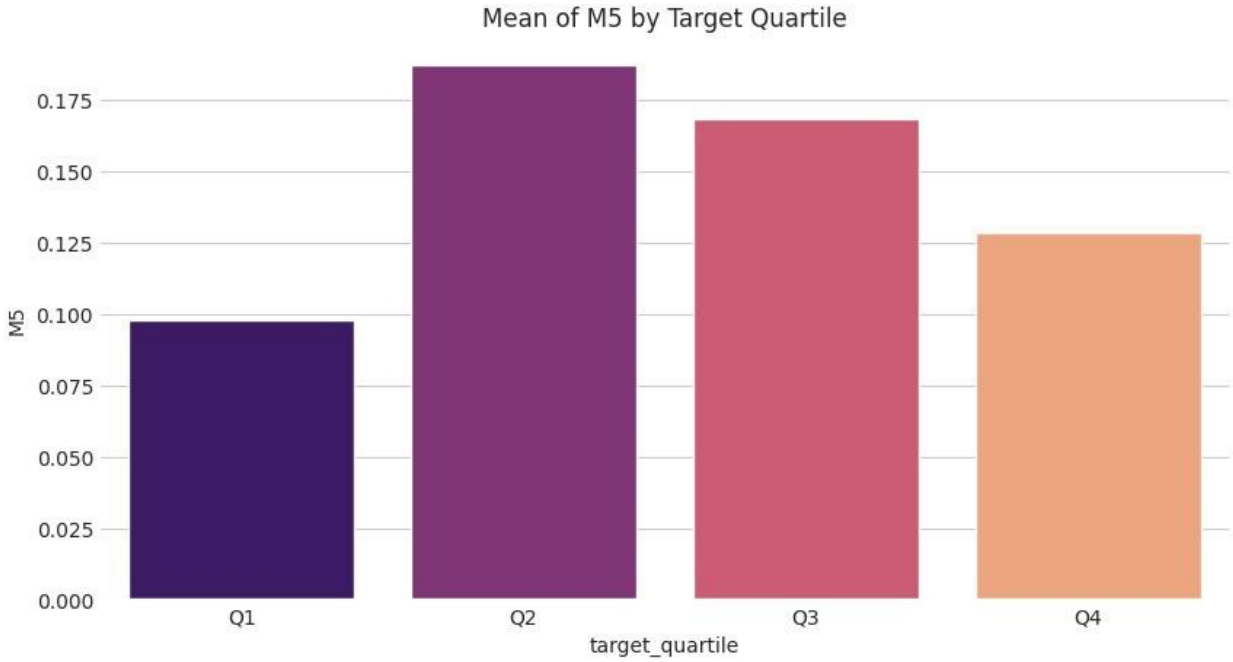

Strip Plot: M5 vs Target Quartiles
M5
6
4
2
0
−2
Q1
Q2
Q3
Q4
target_quartile


Scatter: I5 vs Target colored by I9
market_forward_excess_returns
0.04
0.03
0.02
0.01
0.00
−0.01
−0.02
−0.03
−0.04
−4
−2
0
2
4
I5
I9
4
2
0
−2
−4


KDE + Rug: E8
Density
0.0
0.1
0.2
0.3
0.4
0.5
−5
0
5
10
15
20
25
30
35
E8


Hexbin: Feature vs Target
market_forward_excess_returns
0.04
0.03
0.02
0.01
0.00
−0.01
−0.02
−0.03
−0.04
−20
−15
−10
−5
0
E7
Count
1600
1400
1200
1000
800
600
400
200
0


Rolling Median of Target (20-period)
Median
0.01
0.00
−0.01
−0.02
−0.03
0
2000
4000
6000
8000
Date


Mean of M5 by Target Quartile
M5
0.175
0.150
0.125
0.100
0.075
0.050
0.025
0.000
Q1
Q2
Q3
Q4
target_quartile

```
 10 Mini EDA Plots Completed Successfully!
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import mean_squared_error, mean_absolute_error,
r2_score
```

```
from sklearn.model_selection import train_test_split
import tensorflow as tf

# ---------------------------
# 1) Load Data
# ---------------------------train
=
pd.read_csv('/kaggle/input/hull-tactical-market-prediction/train.csv')
test =
pd.read_csv('/kaggle/input/hull-tactical-market-prediction/test.csv')

# Automatically detect target column if
'market_forward_excess_returns' in train.columns:     TARGET =
'market_forward_excess_returns' elif 'forward_returns' in
train.columns:     TARGET = 'forward_returns' else:     raise
ValueError("No known target column found in train dataset.")
print(f"Using target column: {TARGET}")

# ---------------------------
# 2) Preprocessing
# ---------------------------
# Fill missing values
train.fillna(method='ffill', inplace=True)
train.fillna(method='bfill', inplace=True)
test.fillna(method='ffill', inplace=True)
test.fillna(method='bfill', inplace=True)

# Feature selection (exclude target and date)
train_features = [c for c in train.columns if c not in ['date_id',
TARGET]]
# Only keep features common in both train and test features =

[f for f in train_features if f in test.columns]

print(f"Using {len(features)} common features for modeling.")

# Scale features
scaler = StandardScaler()
X_train = scaler.fit_transform(train[features])
X_test = scaler.transform(test[features])
y_train = train[TARGET].values
y_test = test[TARGET].values if TARGET in test.columns else None

# Optional: split train into train/validation
X_train_sub, X_val, y_train_sub, y_val = train_test_split(
    X_train, y_train, test_size=0.2, random_state=42
)
```

```
# --------------------------
# 3) Build Custom Deep Learning Model
# --------------------------model
= tf.keras.Sequential([
    tf.keras.layers.Dense(256, activation='relu',
input_shape=(X_train.shape[1],)),
tf.keras.layers.Dropout(0.3),
    tf.keras.layers.Dense(128, activation='relu'),
tf.keras.layers.Dropout(0.2),
    tf.keras.layers.Dense(64, activation='relu'),
tf.keras.layers.Dense(1)
])

model.compile(
    optimizer=tf.keras.optimizers.Adam(learning_rate=0.001),
loss='mean_squared_error',     metrics=['mae']
)

model.summary()

# --------------------------
# 4) Train the Model
# --------------------------history
= model.fit(
    X_train_sub, y_train_sub,
validation_data=(X_val, y_val),
    epochs=200,         # increase for higher accuracy
batch_size=32,     verbose=1
)

# --------------------------
# 5) Evaluate Model
# --------------------------if y_test
is not None:     y_pred_test =
model.predict(X_test)
    print("Test MAE:", mean_absolute_error(y_test, y_pred_test))
print("Test MSE:", mean_squared_error(y_test, y_pred_test))
print("Test R2 Score:", r2_score(y_test, y_pred_test))

# --------------------------
# 6) Training History Plot
# --------------------------plt.figure(figsize=(12,5))
plt.plot(history.history['loss'], label='Train Loss')
plt.plot(history.history['val_loss'], label='Validation Loss')
plt.xlabel('Epochs')
plt.ylabel('MSE Loss')
plt.title('Training vs Validation Loss')
```

```
plt.legend()
plt.show()

plt.figure(figsize=(12,5))
plt.plot(history.history['mae'], label='Train MAE')
plt.plot(history.history['val_mae'], label='Validation MAE')
plt.xlabel('Epochs')
plt.ylabel('MAE')
plt.title('Training vs Validation MAE')
plt.legend() plt.show()

# ---------------------------
# 7) Predictions & Submission # ---------
------------------test_predictions =
model.predict(X_test) submission =
pd.DataFrame({     'date_id':
test['date_id'],
    'predicted': test_predictions.flatten()
})
submission.to_csv('submission.csv', index=False)

print(" Model training, evaluation, and submission complete!")

2025-09-20 15:30:55.211747: E
external/local_xla/xla/stream_executor/cuda/cuda_fft.cc:477] Unable to
register cuFFT factory: Attempting to register factory for plugin
cuFFT when one has already been registered
WARNING: All log messages before absl::InitializeLog() is called are
written to STDERR
E0000 00:00:1758382255.502272      36 cuda_dnn.cc:8310] Unable to
register cuDNN factory: Attempting to register factory for plugin
cuDNN when one has already been registered
E0000 00:00:1758382255.578486      36 cuda_blas.cc:1418] Unable to
register cuBLAS factory: Attempting to register factory for plugin
cuBLAS when one has already been registered

Using target column: market_forward_excess_returns
Using 94 common features for modeling.

2025-09-20 15:31:13.238381: E
external/local_xla/xla/stream_executor/cuda/cuda_driver.cc:152] failed
call to cuInit: INTERNAL: CUDA error: Failed call to cuInit: UNKNOWN
ERROR (303)

Model: "sequential"

□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□
□□□□□□
□ Layer (type)                    □ Output Shape              □
Param # □
```

```
□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□
□□□□□□□
│ dense (Dense)                    │ (None, 256)               │
24,320 │
├──────────────────────────────────┼───────────────────────────┼───────────
──────┤
│ dropout (Dropout)                │ (None, 256)               │
0 │
├──────────────────────────────────┼───────────────────────────┼───────────
──────┤
│ dense_1 (Dense)                  │ (None, 128)               │
32,896 │
├──────────────────────────────────┼───────────────────────────┼───────────
──────┤
│ dropout_1 (Dropout)              │ (None, 128)               │
0 │
├──────────────────────────────────┼───────────────────────────┼───────────
──────┤
│ dense_2 (Dense)                  │ (None, 64)                │
8,256 │
├──────────────────────────────────┼───────────────────────────┼───────────
──────┤
│ dense_3 (Dense)                  │ (None, 1)                 │
65 │
└──────────────────────────────────┴───────────────────────────┴───────────
──────┘

 Total params: 65,537 (256.00 KB)

 Trainable params: 65,537 (256.00 KB)  Non-

trainable params: 0 (0.00 B)

Epoch 1/200
225/225 □□□□□□□□□□□□□□□□□□□□ 4s 6ms/step - loss: 0.0713 - mae: 0.1717
- val_loss: 6.9737e-04 - val_mae: 0.0199
Epoch 2/200
225/225 □□□□□□□□□□□□□□□□□□□□ 1s 4ms/step - loss: 0.0027 - mae: 0.0384
- val_loss: 2.7479e-04 - val_mae: 0.0124
Epoch 3/200
225/225 □□□□□□□□□□□□□□□□□□□□ 1s 4ms/step - loss: 0.0012 - mae: 0.0249
- val_loss: 1.8320e-04 - val_mae: 0.0102
Epoch 4/200 225/225 □□□□□□□□□□□□□□□□□□□□ 1s 4ms/step - loss:
6.5544e-04 - mae:
0.0186 - val_loss: 1.4851e-04 - val_mae: 0.0089 Epoch 5/200
225/225 □□□□□□□□□□□□□□□□□□□□ 1s 4ms/step - loss: 4.2889e-04 - mae:
0.0147 - val_loss: 1.3008e-04 - val_mae: 0.0083 Epoch 6/200
225/225 □□□□□□□□□□□□□□□□□□□□ 1s 4ms/step - loss: 2.7411e-04 - mae:
```

```
0.0119 - val_loss: 1.4110e-04 - val_mae: 0.0089 Epoch 7/200
225/225 □□□□□□□□□□□□□□□□□□□□ 1s 4ms/step - loss: 2.5342e-04 - mae:
0.0113 - val_loss: 1.4766e-04 - val_mae: 0.0092 Epoch 8/200
225/225 □□□□□□□□□□□□□□□□□□□□ 1s 4ms/step - loss: 1.9999e-04 - mae:
0.0105 - val_loss: 1.2802e-04 - val_mae: 0.0084 Epoch 9/200
225/225 □□□□□□□□□□□□□□□□□□□□ 1s 4ms/step - loss: 1.8797e-04 - mae:
0.0099 - val_loss: 1.1872e-04 - val_mae: 0.0078 Epoch 10/200
225/225 □□□□□□□□□□□□□□□□□□□□ 1s 4ms/step - loss: 1.6716e-04 - mae:
0.0092 - val_loss: 1.1511e-04 - val_mae: 0.0077 Epoch 11/200
225/225 □□□□□□□□□□□□□□□□□□□□ 1s 4ms/step - loss: 1.5286e-04 - mae:
0.0089 - val_loss: 1.1618e-04 - val_mae: 0.0078 Epoch 12/200
225/225 □□□□□□□□□□□□□□□□□□□□ 1s 4ms/step - loss: 1.4760e-04 - mae:
0.0087 - val_loss: 1.1318e-04 - val_mae: 0.0076 Epoch 13/200
225/225 □□□□□□□□□□□□□□□□□□□□ 1s 4ms/step - loss: 1.5589e-04 - mae:
0.0090 - val_loss: 1.1575e-04 - val_mae: 0.0078 Epoch 14/200
225/225 □□□□□□□□□□□□□□□□□□□□ 1s 4ms/step - loss: 1.3840e-04 - mae:
0.0086 - val_loss: 1.1415e-04 - val_mae: 0.0077 Epoch 15/200
225/225 □□□□□□□□□□□□□□□□□□□□ 1s 4ms/step - loss: 1.4531e-04 - mae:
0.0087 - val_loss: 1.3857e-04 - val_mae: 0.0089 Epoch 16/200
225/225 □□□□□□□□□□□□□□□□□□□□ 1s 4ms/step - loss: 1.3363e-04 - mae:
0.0085 - val_loss: 1.3306e-04 - val_mae: 0.0085 Epoch 17/200
225/225 □□□□□□□□□□□□□□□□□□□□ 1s 4ms/step - loss: 1.3702e-04 - mae:
0.0085 - val_loss: 1.1524e-04 - val_mae: 0.0077 Epoch 18/200
225/225 □□□□□□□□□□□□□□□□□□□□ 1s 4ms/step - loss: 1.2898e-04 - mae:
0.0082 - val_loss: 1.1261e-04 - val_mae: 0.0075 Epoch 19/200
225/225 □□□□□□□□□□□□□□□□□□□□ 1s 4ms/step - loss: 1.2721e-04 - mae:
0.0081 - val_loss: 1.1770e-04 - val_mae: 0.0079 Epoch 20/200
225/225 □□□□□□□□□□□□□□□□□□□□ 1s 4ms/step - loss: 1.2636e-04 - mae:
0.0081 - val_loss: 1.9089e-04 - val_mae: 0.0110 Epoch 21/200
225/225 □□□□□□□□□□□□□□□□□□□□ 1s 4ms/step - loss: 1.3586e-04 - mae:
0.0085 - val_loss: 1.1250e-04 - val_mae: 0.0075 Epoch 22/200
225/225 □□□□□□□□□□□□□□□□□□□□ 1s 4ms/step - loss: 1.2035e-04 - mae:
0.0080 - val_loss: 1.1809e-04 - val_mae: 0.0078
Epoch 23/200 225/225 □□□□□□□□□□□□□□□□□□□□ 1s 4ms/step - loss:
1.2682e-04 - mae:
0.0082 - val_loss: 1.1219e-04 - val_mae: 0.0075 Epoch 24/200
225/225 □□□□□□□□□□□□□□□□□□□□ 1s 4ms/step - loss: 1.2649e-04 - mae:
0.0081 - val_loss: 1.1185e-04 - val_mae: 0.0075 Epoch 25/200
225/225 □□□□□□□□□□□□□□□□□□□□ 1s 4ms/step - loss: 1.1743e-04 - mae:
0.0078 - val_loss: 1.1378e-04 - val_mae: 0.0077 Epoch 26/200
225/225 □□□□□□□□□□□□□□□□□□□□ 1s 4ms/step - loss: 1.2508e-04 - mae:
0.0080 - val_loss: 1.1336e-04 - val_mae: 0.0076 Epoch 27/200
225/225 □□□□□□□□□□□□□□□□□□□□ 1s 4ms/step - loss: 1.1590e-04 - mae:
0.0078 - val_loss: 1.1305e-04 - val_mae: 0.0076 Epoch 28/200
225/225 □□□□□□□□□□□□□□□□□□□□ 1s 4ms/step - loss: 1.2456e-04 - mae:
0.0080 - val_loss: 1.1467e-04 - val_mae: 0.0077 Epoch 29/200
225/225 □□□□□□□□□□□□□□□□□□□□ 1s 4ms/step - loss: 1.2336e-04 - mae:
```

0.0080 - val_loss: 1.1384e-04 - val_mae: 0.0076 Epoch 30/200
225/225 □□□□□□□□□□□□□□□□□□□□□□□□ 1s 4ms/step - loss: 1.1896e-04 - mae:
0.0078 - val_loss: 1.2308e-04 - val_mae: 0.0082 Epoch 31/200
225/225 □□□□□□□□□□□□□□□□□□□□□□□□ 1s 4ms/step - loss: 1.1439e-04 - mae:
0.0077 - val_loss: 1.1492e-04 - val_mae: 0.0076 Epoch 32/200
225/225 □□□□□□□□□□□□□□□□□□□□□□□□ 1s 4ms/step - loss: 1.2021e-04 - mae:
0.0078 - val_loss: 1.1237e-04 - val_mae: 0.0075 Epoch 33/200
225/225 □□□□□□□□□□□□□□□□□□□□□□□□ 1s 4ms/step - loss: 1.2190e-04 - mae:
0.0079 - val_loss: 1.1170e-04 - val_mae: 0.0075 Epoch 34/200
225/225 □□□□□□□□□□□□□□□□□□□□□□□□ 1s 4ms/step - loss: 1.1078e-04 - mae:
0.0076 - val_loss: 1.1129e-04 - val_mae: 0.0075 Epoch 35/200
225/225 □□□□□□□□□□□□□□□□□□□□□□□□ 1s 4ms/step - loss: 1.1705e-04 - mae:
0.0078 - val_loss: 1.1405e-04 - val_mae: 0.0077 Epoch 36/200
225/225 □□□□□□□□□□□□□□□□□□□□□□□□ 1s 4ms/step - loss: 1.1532e-04 - mae:
0.0077 - val_loss: 1.1182e-04 - val_mae: 0.0075 Epoch 37/200
225/225 □□□□□□□□□□□□□□□□□□□□□□□□ 1s 4ms/step - loss: 1.1755e-04 - mae:
0.0077 - val_loss: 1.1253e-04 - val_mae: 0.0075 Epoch 38/200
225/225 □□□□□□□□□□□□□□□□□□□□□□□□ 1s 4ms/step - loss: 1.1917e-04 - mae:
0.0079 - val_loss: 1.2542e-04 - val_mae: 0.0082
Epoch 39/200

```
225/225 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - loss: 1.2343e-04 - mae:
0.0079 - val_loss: 1.1183e-04 - val_mae: 0.0075
Epoch 40/200

225/225 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - loss: 1.2205e-04 - mae:
0.0079 - val_loss: 1.1278e-04 - val_mae: 0.0076
Epoch 41/200

225/225 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - loss: 1.1499e-04 - mae:
0.0077 - val_loss: 1.1185e-04 - val_mae: 0.0075
Epoch 42/200

225/225 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - loss: 1.1262e-04 - mae:
0.0076 - val_loss: 1.1249e-04 - val_mae: 0.0075
Epoch 43/200

225/225 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - loss: 1.1130e-04 - mae:
0.0076 - val_loss: 1.1153e-04 - val_mae: 0.0075
Epoch 44/200

225/225 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - loss: 1.1889e-04 - mae:
0.0078 - val_loss: 1.1468e-04 - val_mae: 0.0076
Epoch 45/200

225/225 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - loss: 1.1498e-04 - mae:
0.0078 - val_loss: 1.1108e-04 - val_mae: 0.0075
Epoch 46/200

225/225 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - loss: 1.1282e-04 - mae:
0.0076 - val_loss: 1.1664e-04 - val_mae: 0.0079
Epoch 47/200

225/225 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - loss: 1.1187e-04 - mae:
0.0076 - val_loss: 1.1139e-04 - val_mae: 0.0075
Epoch 48/200

225/225 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - loss: 1.1093e-04 - mae:
0.0075 - val_loss: 1.1235e-04 - val_mae: 0.0076
Epoch 49/200

225/225 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - loss: 1.1156e-04 - mae:
0.0076 - val_loss: 1.1133e-04 - val_mae: 0.0075
Epoch 50/200

225/225 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - loss: 1.0864e-04 - mae:
0.0074 - val_loss: 1.1396e-04 - val_mae: 0.0077
Epoch 51/200

225/225 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - loss: 1.0724e-04 - mae:
```

```
0.0074 - val_loss: 1.1316e-04 - val_mae: 0.0076
Epoch 52/200
225/225 □□□□□□□□□□□□□□□□□□□□□□□□ 1s 4ms/step - loss: 1.1314e-04 - mae:
0.0076 - val_loss: 1.1170e-04 - val_mae: 0.0075
Epoch 53/200
225/225 □□□□□□□□□□□□□□□□□□□□□□□□ 1s 4ms/step - loss: 1.0920e-04 - mae:
0.0075 - val_loss: 1.1961e-04 - val_mae: 0.0080
Epoch 54/200
225/225 □□□□□□□□□□□□□□□□□□□□□□□□ 1s 4ms/step - loss: 1.1717e-04 - mae:
0.0077 - val_loss: 1.1158e-04 - val_mae: 0.0075
Epoch 55/200
225/225 □□□□□□□□□□□□□□□□□□□□□□□□ 1s 4ms/step - loss: 1.1058e-04 - mae:
```

```
0.0075 - val_loss: 1.1733e-04 - val_mae: 0.0077 Epoch 56/200
225/225 □□□□□□□□□□□□□□□□□□□□ 1s 5ms/step - loss: 1.1112e-04 - mae:
0.0075 - val_loss: 1.1169e-04 - val_mae: 0.0075 Epoch 57/200
225/225 □□□□□□□□□□□□□□□□□□□□ 1s 4ms/step - loss: 1.1560e-04 - mae:
0.0076 - val_loss: 1.1145e-04 - val_mae: 0.0076 Epoch 58/200
225/225 □□□□□□□□□□□□□□□□□□□□ 1s 4ms/step - loss: 1.1217e-04 - mae:
0.0075 - val_loss: 1.1215e-04 - val_mae: 0.0075 Epoch 59/200
225/225 □□□□□□□□□□□□□□□□□□□□ 1s 4ms/step - loss: 1.1046e-04 - mae:
0.0075 - val_loss: 1.1108e-04 - val_mae: 0.0075 Epoch 60/200
225/225 □□□□□□□□□□□□□□□□□□□□ 1s 4ms/step - loss: 1.1421e-04 - mae:
0.0076 - val_loss: 1.1333e-04 - val_mae: 0.0075 Epoch 61/200
225/225 □□□□□□□□□□□□□□□□□□□□ 1s 4ms/step - loss: 1.1685e-04 - mae:
0.0077 - val_loss: 1.3963e-04 - val_mae: 0.0077 Epoch 62/200
225/225 □□□□□□□□□□□□□□□□□□□□ 1s 4ms/step - loss: 1.1622e-04 - mae:
0.0077 - val_loss: 1.1290e-04 - val_mae: 0.0076 Epoch 63/200
225/225 □□□□□□□□□□□□□□□□□□□□ 1s 5ms/step - loss: 1.0617e-04 - mae:
0.0074 - val_loss: 1.1365e-04 - val_mae: 0.0075 Epoch 64/200
225/225 □□□□□□□□□□□□□□□□□□□□ 1s 5ms/step - loss: 1.1972e-04 - mae:
0.0076 - val_loss: 1.1239e-04 - val_mae: 0.0075 Epoch 65/200
225/225 □□□□□□□□□□□□□□□□□□□□ 1s 4ms/step - loss: 1.0896e-04 - mae:
0.0074 - val_loss: 1.1710e-04 - val_mae: 0.0076 Epoch 66/200
225/225 □□□□□□□□□□□□□□□□□□□□ 1s 4ms/step - loss: 1.1861e-04 - mae:
0.0076 - val_loss: 1.1208e-04 - val_mae: 0.0075 Epoch 67/200
225/225 □□□□□□□□□□□□□□□□□□□□ 1s 4ms/step - loss: 1.0639e-04 - mae:
0.0073 - val_loss: 1.1433e-04 - val_mae: 0.0076 Epoch 68/200
225/225 □□□□□□□□□□□□□□□□□□□□ 1s 5ms/step - loss: 1.1153e-04 - mae:
0.0075 - val_loss: 1.1236e-04 - val_mae: 0.0075 Epoch 69/200
225/225 □□□□□□□□□□□□□□□□□□□□ 1s 4ms/step - loss: 1.1303e-04 - mae:
0.0075 - val_loss: 1.1704e-04 - val_mae: 0.0077 Epoch 70/200
225/225 □□□□□□□□□□□□□□□□□□□□ 1s 4ms/step - loss: 1.0761e-04 - mae:
0.0074 - val_loss: 1.1481e-04 - val_mae: 0.0076 Epoch 71/200
225/225 □□□□□□□□□□□□□□□□□□□□ 1s 4ms/step - loss: 1.0820e-04 - mae:
0.0074 - val_loss: 1.1513e-04 - val_mae: 0.0077
Epoch 72/200 225/225 □□□□□□□□□□□□□□□□□□□□ 1s 4ms/step - loss:
1.1321e-04 - mae:
0.0076 - val_loss: 1.1428e-04 - val_mae: 0.0077 Epoch 73/200
225/225 □□□□□□□□□□□□□□□□□□□□ 1s 4ms/step - loss: 1.1093e-04 - mae:
0.0076 - val_loss: 1.1339e-04 - val_mae: 0.0075 Epoch 74/200
225/225 □□□□□□□□□□□□□□□□□□□□ 1s 4ms/step - loss: 1.0641e-04 - mae:
0.0073 - val_loss: 1.1222e-04 - val_mae: 0.0076 Epoch 75/200
225/225 □□□□□□□□□□□□□□□□□□□□ 1s 4ms/step - loss: 1.1127e-04 - mae:
0.0075 - val_loss: 1.1427e-04 - val_mae: 0.0075 Epoch 76/200
225/225 □□□□□□□□□□□□□□□□□□□□ 1s 4ms/step - loss: 1.1292e-04 - mae:
0.0076 - val_loss: 1.1200e-04 - val_mae: 0.0075 Epoch 77/200
225/225 □□□□□□□□□□□□□□□□□□□□ 1s 4ms/step - loss: 1.0816e-04 - mae:
0.0074 - val_loss: 1.1279e-04 - val_mae: 0.0075 Epoch 78/200
225/225 □□□□□□□□□□□□□□□□□□□□ 1s 4ms/step - loss: 1.0989e-04 - mae:
```

0.0075 - val_loss: 1.1239e-04 - val_mae: 0.0076 Epoch 79/200
225/225 □□□□□□□□□□□□□□□□□□□□□□□□ 1s 4ms/step - loss: 1.1179e-04 - mae:
0.0075 - val_loss: 1.1315e-04 - val_mae: 0.0076 Epoch 80/200
225/225 □□□□□□□□□□□□□□□□□□□□□□□□ 1s 4ms/step - loss: 1.1233e-04 - mae:
0.0076 - val_loss: 1.1205e-04 - val_mae: 0.0075 Epoch 81/200
225/225 □□□□□□□□□□□□□□□□□□□□□□□□ 1s 4ms/step - loss: 1.0196e-04 - mae:
0.0072 - val_loss: 1.1424e-04 - val_mae: 0.0076 Epoch 82/200
225/225 □□□□□□□□□□□□□□□□□□□□□□□□ 1s 4ms/step - loss: 1.0770e-04 - mae:
0.0074 - val_loss: 1.1367e-04 - val_mae: 0.0076 Epoch 83/200
225/225 □□□□□□□□□□□□□□□□□□□□□□□□ 1s 4ms/step - loss: 1.0552e-04 - mae:
0.0073 - val_loss: 1.1414e-04 - val_mae: 0.0076 Epoch 84/200
225/225 □□□□□□□□□□□□□□□□□□□□□□□□ 1s 4ms/step - loss: 1.1185e-04 - mae:
0.0075 - val_loss: 1.1282e-04 - val_mae: 0.0076 Epoch 85/200
225/225 □□□□□□□□□□□□□□□□□□□□□□□□ 1s 4ms/step - loss: 1.1193e-04 - mae:
0.0075 - val_loss: 1.1972e-04 - val_mae: 0.0079 Epoch 86/200
225/225 □□□□□□□□□□□□□□□□□□□□□□□□ 1s 4ms/step - loss: 1.2397e-04 - mae:
0.0076 - val_loss: 1.1571e-04 - val_mae: 0.0076 Epoch 87/200
225/225 □□□□□□□□□□□□□□□□□□□□□□□□ 1s 4ms/step - loss: 1.0997e-04 - mae:
0.0075 - val_loss: 1.1512e-04 - val_mae: 0.0075
Epoch 88/200

```
225/225 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - loss: 1.0780e-04 - mae:
0.0073 - val_loss: 1.2055e-04 - val_mae: 0.0077
Epoch 89/200

225/225 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - loss: 1.1094e-04 - mae:
0.0075 - val_loss: 1.1609e-04 - val_mae: 0.0076
Epoch 90/200

225/225 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - loss: 1.0558e-04 - mae:
0.0073 - val_loss: 1.1609e-04 - val_mae: 0.0078
Epoch 91/200

225/225 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - loss: 1.1983e-04 - mae:
0.0077 - val_loss: 1.1562e-04 - val_mae: 0.0076
Epoch 92/200

225/225 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - loss: 1.0933e-04 - mae:
0.0075 - val_loss: 1.1376e-04 - val_mae: 0.0076
Epoch 93/200

225/225 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - loss: 1.0645e-04 - mae:
0.0074 - val_loss: 1.1326e-04 - val_mae: 0.0075
Epoch 94/200

225/225 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - loss: 1.0621e-04 - mae:
0.0074 - val_loss: 1.1438e-04 - val_mae: 0.0075
Epoch 95/200

225/225 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - loss: 1.0593e-04 - mae:
0.0073 - val_loss: 1.1533e-04 - val_mae: 0.0076
Epoch 96/200

225/225 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - loss: 1.0538e-04 - mae:
0.0072 - val_loss: 1.1471e-04 - val_mae: 0.0076
Epoch 97/200

225/225 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - loss: 1.0699e-04 - mae:
0.0074 - val_loss: 1.1516e-04 - val_mae: 0.0078
Epoch 98/200

225/225 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - loss: 1.0708e-04 - mae:
0.0074 - val_loss: 1.1308e-04 - val_mae: 0.0075
Epoch 99/200

225/225 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - loss: 1.0502e-04 - mae:
0.0073 - val_loss: 1.1408e-04 - val_mae: 0.0076
Epoch 100/200

225/225 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - loss: 1.0393e-04 - mae:
```

```
0.0073 - val_loss: 1.1525e-04 - val_mae: 0.0076
Epoch 101/200

225/225 □□□□□□□□□□□□□□□□□□□□□□□□ 1s 4ms/step - loss: 1.0192e-04 - mae:
0.0073 - val_loss: 1.1664e-04 - val_mae: 0.0076
Epoch 102/200

225/225 □□□□□□□□□□□□□□□□□□□□□□□□ 1s 4ms/step - loss: 1.0320e-04 - mae:
0.0073 - val_loss: 1.1765e-04 - val_mae: 0.0076
Epoch 103/200

225/225 □□□□□□□□□□□□□□□□□□□□□□□□ 1s 4ms/step - loss: 1.0554e-04 - mae:
0.0073 - val_loss: 1.2321e-04 - val_mae: 0.0082
Epoch 104/200

225/225 □□□□□□□□□□□□□□□□□□□□□□□□ 1s 4ms/step - loss: 1.0520e-04 - mae:
```

```
0.0074 - val_loss: 1.1492e-04 - val_mae: 0.0077 Epoch 105/200
225/225 □□□□□□□□□□□□□□□□□□□□ 1s 4ms/step - loss: 1.0737e-04 - mae:
0.0074 - val_loss: 1.1572e-04 - val_mae: 0.0076 Epoch 106/200
225/225 □□□□□□□□□□□□□□□□□□□□ 1s 4ms/step - loss: 1.0313e-04 - mae:
0.0072 - val_loss: 1.2154e-04 - val_mae: 0.0081 Epoch 107/200
225/225 □□□□□□□□□□□□□□□□□□□□ 1s 4ms/step - loss: 1.0338e-04 - mae:
0.0073 - val_loss: 1.1545e-04 - val_mae: 0.0076 Epoch 108/200
225/225 □□□□□□□□□□□□□□□□□□□□ 1s 4ms/step - loss: 1.0081e-04 - mae:
0.0072 - val_loss: 1.1421e-04 - val_mae: 0.0076 Epoch 109/200
225/225 □□□□□□□□□□□□□□□□□□□□ 1s 4ms/step - loss: 1.0538e-04 - mae:
0.0073 - val_loss: 1.1317e-04 - val_mae: 0.0075 Epoch 110/200
225/225 □□□□□□□□□□□□□□□□□□□□ 1s 4ms/step - loss: 1.0159e-04 - mae:
0.0072 - val_loss: 1.1380e-04 - val_mae: 0.0076 Epoch 111/200
225/225 □□□□□□□□□□□□□□□□□□□□ 1s 4ms/step - loss: 1.0300e-04 - mae:
0.0073 - val_loss: 1.1704e-04 - val_mae: 0.0076 Epoch 112/200
225/225 □□□□□□□□□□□□□□□□□□□□ 1s 4ms/step - loss: 1.0637e-04 - mae:
0.0074 - val_loss: 1.1420e-04 - val_mae: 0.0076 Epoch 113/200
225/225 □□□□□□□□□□□□□□□□□□□□ 1s 4ms/step - loss: 1.0136e-04 - mae:
0.0072 - val_loss: 1.1726e-04 - val_mae: 0.0076 Epoch 114/200
225/225 □□□□□□□□□□□□□□□□□□□□ 1s 4ms/step - loss: 1.0622e-04 - mae:
0.0073 - val_loss: 1.1516e-04 - val_mae: 0.0077 Epoch 115/200
225/225 □□□□□□□□□□□□□□□□□□□□ 1s 4ms/step - loss: 1.0226e-04 - mae:
0.0073 - val_loss: 1.1624e-04 - val_mae: 0.0076 Epoch 116/200
225/225 □□□□□□□□□□□□□□□□□□□□ 1s 4ms/step - loss: 9.7942e-05 - mae:
0.0071 - val_loss: 1.1798e-04 - val_mae: 0.0076 Epoch 117/200
225/225 □□□□□□□□□□□□□□□□□□□□ 1s 4ms/step - loss: 1.0613e-04 - mae:
0.0073 - val_loss: 1.1507e-04 - val_mae: 0.0077 Epoch 118/200
225/225 □□□□□□□□□□□□□□□□□□□□ 1s 4ms/step - loss: 1.0432e-04 - mae:
0.0073 - val_loss: 1.1544e-04 - val_mae: 0.0076 Epoch 119/200
225/225 □□□□□□□□□□□□□□□□□□□□ 1s 4ms/step - loss: 9.8177e-05 - mae:
0.0072 - val_loss: 1.1715e-04 - val_mae: 0.0076 Epoch 120/200
225/225 □□□□□□□□□□□□□□□□□□□□ 1s 4ms/step - loss: 1.0292e-04 - mae:
0.0072 - val_loss: 1.1568e-04 - val_mae: 0.0077
Epoch 121/200 225/225 □□□□□□□□□□□□□□□□□□□□ 1s 4ms/step - loss:
1.0290e-04 - mae:
0.0072 - val_loss: 1.1689e-04 - val_mae: 0.0078 Epoch 122/200
225/225 □□□□□□□□□□□□□□□□□□□□ 1s 4ms/step - loss: 1.0151e-04 - mae:
0.0073 - val_loss: 1.1535e-04 - val_mae: 0.0076 Epoch 123/200
225/225 □□□□□□□□□□□□□□□□□□□□ 1s 4ms/step - loss: 9.9759e-05 - mae:
0.0072 - val_loss: 1.1550e-04 - val_mae: 0.0078 Epoch 124/200
225/225 □□□□□□□□□□□□□□□□□□□□ 1s 4ms/step - loss: 9.7370e-05 - mae:
0.0071 - val_loss: 1.1527e-04 - val_mae: 0.0076 Epoch 125/200
225/225 □□□□□□□□□□□□□□□□□□□□ 1s 4ms/step - loss: 1.0028e-04 - mae:
0.0072 - val_loss: 1.1514e-04 - val_mae: 0.0077 Epoch 126/200
225/225 □□□□□□□□□□□□□□□□□□□□ 1s 4ms/step - loss: 1.0311e-04 - mae:
0.0073 - val_loss: 1.1433e-04 - val_mae: 0.0076 Epoch 127/200
225/225 □□□□□□□□□□□□□□□□□□□□ 1s 4ms/step - loss: 1.0009e-04 - mae:
```

0.0072 - val_loss: 1.1342e-04 - val_mae: 0.0076 Epoch 128/200
225/225 □□□□□□□□□□□□□□□□□□□□□□ 1s 4ms/step - loss: 1.0257e-04 - mae:
0.0072 - val_loss: 1.1443e-04 - val_mae: 0.0076 Epoch 129/200
225/225 □□□□□□□□□□□□□□□□□□□□□□ 1s 4ms/step - loss: 1.0373e-04 - mae:
0.0072 - val_loss: 1.1352e-04 - val_mae: 0.0076 Epoch 130/200
225/225 □□□□□□□□□□□□□□□□□□□□□□ 1s 4ms/step - loss: 1.0086e-04 - mae:
0.0072 - val_loss: 1.1465e-04 - val_mae: 0.0076 Epoch 131/200
225/225 □□□□□□□□□□□□□□□□□□□□□□ 1s 4ms/step - loss: 1.0315e-04 - mae:
0.0072 - val_loss: 1.1700e-04 - val_mae: 0.0078 Epoch 132/200
225/225 □□□□□□□□□□□□□□□□□□□□□□ 1s 4ms/step - loss: 1.0001e-04 - mae:
0.0072 - val_loss: 1.1492e-04 - val_mae: 0.0076 Epoch 133/200
225/225 □□□□□□□□□□□□□□□□□□□□□□ 1s 4ms/step - loss: 9.9831e-05 - mae:
0.0071 - val_loss: 1.1560e-04 - val_mae: 0.0077 Epoch 134/200
225/225 □□□□□□□□□□□□□□□□□□□□□□ 1s 4ms/step - loss: 9.9201e-05 - mae:
0.0072 - val_loss: 1.1371e-04 - val_mae: 0.0076 Epoch 135/200
225/225 □□□□□□□□□□□□□□□□□□□□□□ 1s 4ms/step - loss: 9.6346e-05 - mae:
0.0070 - val_loss: 1.1766e-04 - val_mae: 0.0078 Epoch 136/200
225/225 □□□□□□□□□□□□□□□□□□□□□□ 1s 4ms/step - loss: 9.8955e-05 - mae:
0.0071 - val_loss: 1.1393e-04 - val_mae: 0.0076
Epoch 137/200

```
225/225 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - loss: 9.9055e-05 - mae:
0.0071 - val_loss: 1.1655e-04 - val_mae: 0.0078
Epoch 138/200

225/225 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - loss: 9.4968e-05 - mae:
0.0071 - val_loss: 1.1932e-04 - val_mae: 0.0078
Epoch 139/200

225/225 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - loss: 9.9474e-05 - mae:
0.0072 - val_loss: 1.1435e-04 - val_mae: 0.0076
Epoch 140/200

225/225 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - loss: 9.5084e-05 - mae:
0.0070 - val_loss: 1.2034e-04 - val_mae: 0.0080
Epoch 141/200

225/225 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - loss: 9.7320e-05 - mae:
0.0071 - val_loss: 1.1533e-04 - val_mae: 0.0076
Epoch 142/200

225/225 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - loss: 9.9037e-05 - mae:
0.0071 - val_loss: 1.1642e-04 - val_mae: 0.0076
Epoch 143/200

225/225 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - loss: 9.2697e-05 - mae:
0.0069 - val_loss: 1.1523e-04 - val_mae: 0.0076
Epoch 144/200

225/225 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - loss: 9.8729e-05 - mae:
0.0072 - val_loss: 1.1848e-04 - val_mae: 0.0076
Epoch 145/200

225/225 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - loss: 9.5133e-05 - mae:
0.0070 - val_loss: 1.1433e-04 - val_mae: 0.0077
Epoch 146/200

225/225 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - loss: 1.0072e-04 - mae:
0.0072 - val_loss: 1.1799e-04 - val_mae: 0.0079
Epoch 147/200

225/225 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - loss: 9.5632e-05 - mae:
0.0070 - val_loss: 1.1091e-04 - val_mae: 0.0075
Epoch 148/200

225/225 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - loss: 9.4085e-05 - mae:
0.0070 - val_loss: 1.1365e-04 - val_mae: 0.0076
Epoch 149/200

225/225 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - loss: 9.6971e-05 - mae:
```

```
0.0071 - val_loss: 1.1683e-04 - val_mae: 0.0077
Epoch 150/200

225/225 □□□□□□□□□□□□□□□□□□□□□□□□ 1s 4ms/step - loss: 9.6624e-05 - mae:
0.0071 - val_loss: 1.1494e-04 - val_mae: 0.0076
Epoch 151/200

225/225 □□□□□□□□□□□□□□□□□□□□□□□□ 1s 4ms/step - loss: 9.5471e-05 - mae:
0.0070 - val_loss: 1.1705e-04 - val_mae: 0.0078
Epoch 152/200

225/225 □□□□□□□□□□□□□□□□□□□□□□□□ 1s 4ms/step - loss: 9.6322e-05 - mae:
0.0071 - val_loss: 1.1693e-04 - val_mae: 0.0079
Epoch 153/200

225/225 □□□□□□□□□□□□□□□□□□□□□□□□ 1s 4ms/step - loss: 9.7862e-05 - mae:
```

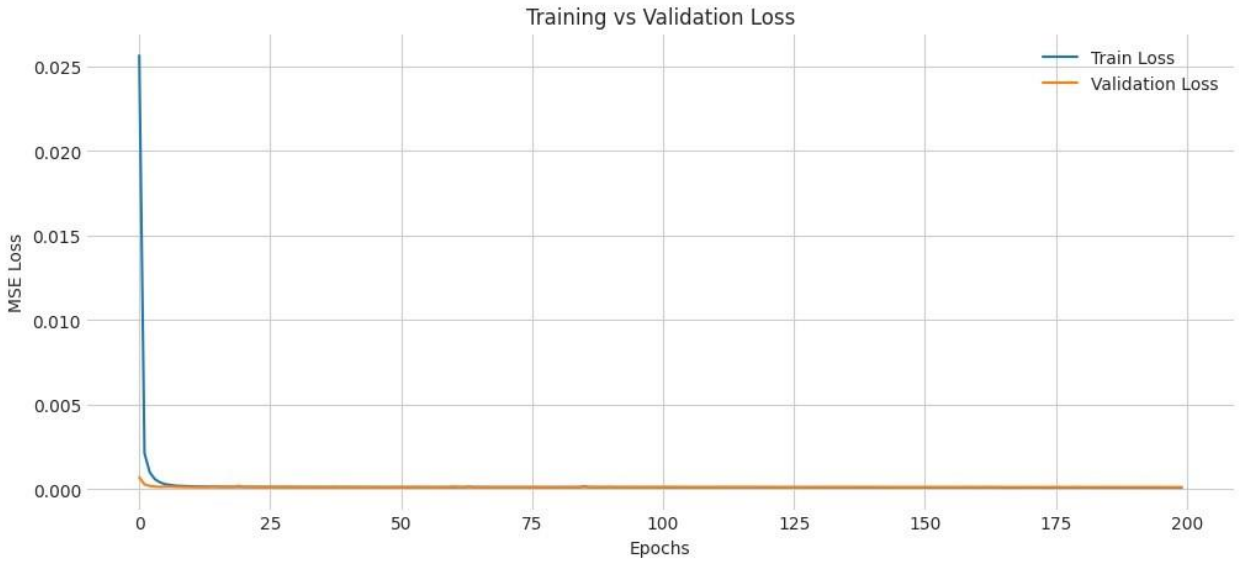
```
0.0071 - val_loss: 1.1949e-04 - val_mae: 0.0077
Epoch 154/200

225/225 □□□□□□□□□□□□□□□□□□□□□□□□ 1s 4ms/step - loss: 9.7851e-05 - mae:
0.0071 - val_loss: 1.1691e-04 - val_mae: 0.0078
Epoch 155/200

225/225 □□□□□□□□□□□□□□□□□□□□□□□□ 1s 4ms/step - loss: 9.7761e-05 - mae:
0.0070 - val_loss: 1.1455e-04 - val_mae: 0.0076
Epoch 156/200

225/225 □□□□□□□□□□□□□□□□□□□□□□□□ 1s 4ms/step - loss: 9.6337e-05 - mae:
0.0071 - val_loss: 1.1645e-04 - val_mae: 0.0076
Epoch 157/200

225/225 □□□□□□□□□□□□□□□□□□□□□□□□ 1s 4ms/step - loss: 9.2601e-05 - mae:
0.0069 - val_loss: 1.1459e-04 - val_mae: 0.0077
Epoch 158/200

225/225 □□□□□□□□□□□□□□□□□□□□□□□□ 1s 4ms/step - loss: 9.1046e-05 - mae:
0.0069 - val_loss: 1.1607e-04 - val_mae: 0.0077
Epoch 159/200

225/225 □□□□□□□□□□□□□□□□□□□□□□□□ 1s 4ms/step - loss: 1.0824e-04 - mae:
0.0072 - val_loss: 1.1450e-04 - val_mae: 0.0076
Epoch 160/200

225/225 □□□□□□□□□□□□□□□□□□□□□□□□ 1s 4ms/step - loss: 9.2987e-05 - mae:
0.0069 - val_loss: 1.1789e-04 - val_mae: 0.0078
Epoch 161/200

225/225 □□□□□□□□□□□□□□□□□□□□□□□□ 1s 4ms/step - loss: 9.2719e-05 - mae:
0.0069 - val_loss: 1.1399e-04 - val_mae: 0.0076
Epoch 162/200

225/225 □□□□□□□□□□□□□□□□□□□□□□□□ 1s 4ms/step - loss: 1.0634e-04 - mae:
0.0072 - val_loss: 1.1588e-04 - val_mae: 0.0076
Epoch 163/200

225/225 □□□□□□□□□□□□□□□□□□□□□□□□ 1s 4ms/step - loss: 9.5730e-05 - mae:
0.0071 - val_loss: 1.2080e-04 - val_mae: 0.0077
Epoch 164/200

225/225 □□□□□□□□□□□□□□□□□□□□□□□□ 1s 4ms/step - loss: 9.9715e-05 - mae:
0.0071 - val_loss: 1.1862e-04 - val_mae: 0.0079
Epoch 165/200

225/225 □□□□□□□□□□□□□□□□□□□□□□□□ 1s 4ms/step - loss: 9.4314e-05 - mae:
0.0070 - val_loss: 1.1736e-04 - val_mae: 0.0078
```

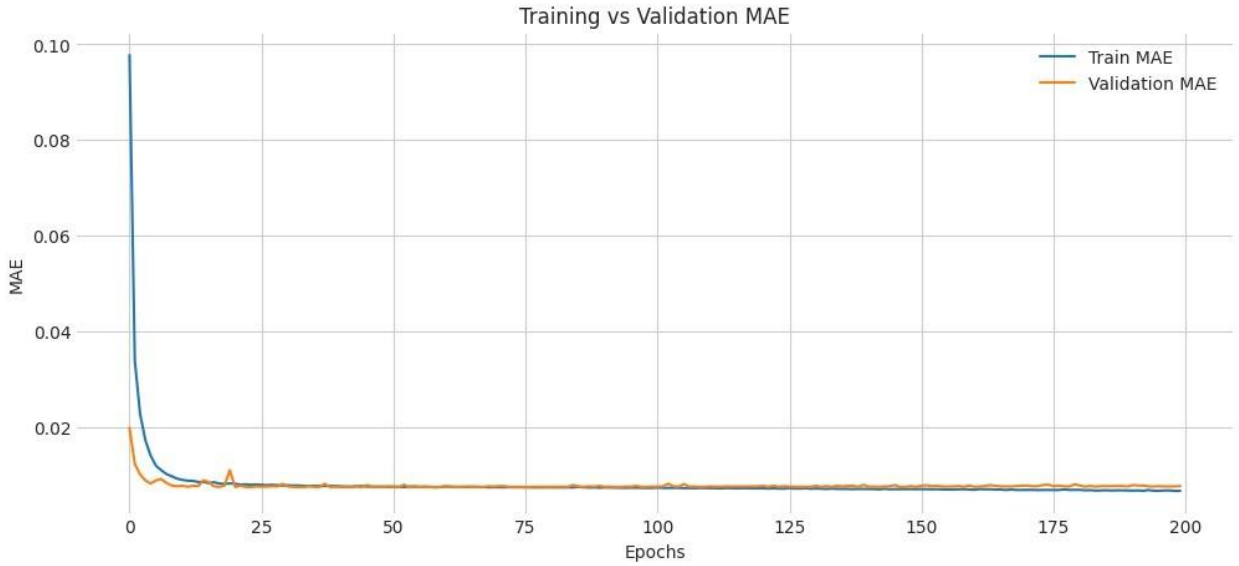
```
Epoch 166/200
225/225 □□□□□□□□□□□□□□□□□□□□□□□□ 1s 4ms/step - loss: 9.2198e-05 - mae:
0.0070 - val_loss: 1.1850e-04 - val_mae: 0.0077
Epoch 167/200
225/225 □□□□□□□□□□□□□□□□□□□□□□□□ 1s 4ms/step - loss: 9.2927e-05 - mae:
0.0070 - val_loss: 1.1807e-04 - val_mae: 0.0076
Epoch 168/200
225/225 □□□□□□□□□□□□□□□□□□□□□□□□ 1s 4ms/step - loss: 9.1284e-05 - mae:
0.0069 - val_loss: 1.1658e-04 - val_mae: 0.0076
Epoch 169/200
225/225 □□□□□□□□□□□□□□□□□□□□□□□□ 1s 4ms/step - loss: 9.3077e-05 - mae:
0.0069 - val_loss: 1.1805e-04 - val_mae: 0.0077
Epoch 170/200
```

```
225/225 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - loss: 9.7103e-05 - mae:
0.0070 - val_loss: 1.1818e-04 - val_mae: 0.0078
Epoch 171/200

225/225 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - loss: 9.1715e-05 - mae:
0.0070 - val_loss: 1.1788e-04 - val_mae: 0.0078
Epoch 172/200

225/225 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - loss: 9.6167e-05 - mae:
0.0071 - val_loss: 1.1911e-04 - val_mae: 0.0077
Epoch 173/200

225/225 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - loss: 9.1554e-05 - mae:
0.0069 - val_loss: 1.2059e-04 - val_mae: 0.0077
Epoch 174/200

225/225 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - loss: 9.2970e-05 - mae:
0.0069 - val_loss: 1.2037e-04 - val_mae: 0.0079
Epoch 175/200

225/225 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - loss: 9.1173e-05 - mae:
0.0069 - val_loss: 1.2027e-04 - val_mae: 0.0080
Epoch 176/200

225/225 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - loss: 9.1474e-05 - mae:
0.0069 - val_loss: 1.2022e-04 - val_mae: 0.0077
Epoch 177/200

225/225 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - loss: 8.9926e-05 - mae:
0.0068 - val_loss: 1.1982e-04 - val_mae: 0.0078
Epoch 178/200

225/225 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - loss: 9.1547e-05 - mae:
0.0069 - val_loss: 1.1987e-04 - val_mae: 0.0077
Epoch 179/200

225/225 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - loss: 9.1235e-05 - mae:
0.0070 - val_loss: 1.1746e-04 - val_mae: 0.0076
Epoch 180/200

225/225 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - loss: 9.2821e-05 - mae:
0.0069 - val_loss: 1.2424e-04 - val_mae: 0.0081
Epoch 181/200

225/225 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - loss: 9.4630e-05 - mae:
0.0070 - val_loss: 1.1877e-04 - val_mae: 0.0078
Epoch 182/200

225/225 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - loss: 8.8397e-05 - mae:
0.0068 - val_loss: 1.1677e-04 - val_mae: 0.0076
```

```
Epoch 183/200
225/225 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - loss: 9.3368e-05 - mae:
0.0070 - val_loss: 1.1603e-04 - val_mae: 0.0077
Epoch 184/200
225/225 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - loss: 8.5857e-05 - mae:
0.0066 - val_loss: 1.1779e-04 - val_mae: 0.0076
Epoch 185/200
225/225 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - loss: 8.8574e-05 - mae:
0.0068 - val_loss: 1.2101e-04 - val_mae: 0.0077
Epoch 186/200
225/225 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - loss: 9.1897e-05 - mae:
```

```
0.0069 - val_loss: 1.2215e-04 - val_mae: 0.0077 Epoch 187/200
225/225 □□□□□□□□□□□□□□□□□□□□□□□□ 1s 4ms/step - loss: 8.8736e-05 - mae:
0.0069 - val_loss: 1.2091e-04 - val_mae: 0.0077 Epoch 188/200
225/225 □□□□□□□□□□□□□□□□□□□□□□□□ 1s 4ms/step - loss: 9.0538e-05 - mae:
0.0069 - val_loss: 1.1833e-04 - val_mae: 0.0077 Epoch 189/200
225/225 □□□□□□□□□□□□□□□□□□□□□□□□ 1s 4ms/step - loss: 9.0445e-05 - mae:
0.0068 - val_loss: 1.1852e-04 - val_mae: 0.0077 Epoch 190/200
225/225 □□□□□□□□□□□□□□□□□□□□□□□□ 1s 4ms/step - loss: 8.8338e-05 - mae:
0.0068 - val_loss: 1.1920e-04 - val_mae: 0.0076 Epoch 191/200
225/225 □□□□□□□□□□□□□□□□□□□□□□□□ 1s 4ms/step - loss: 8.6783e-05 - mae:
0.0068 - val_loss: 1.2194e-04 - val_mae: 0.0079 Epoch 192/200
225/225 □□□□□□□□□□□□□□□□□□□□□□□□ 1s 4ms/step - loss: 8.6548e-05 - mae:
0.0068 - val_loss: 1.2239e-04 - val_mae: 0.0078 Epoch 193/200
225/225 □□□□□□□□□□□□□□□□□□□□□□□□ 1s 4ms/step - loss: 9.0292e-05 - mae:
0.0068 - val_loss: 1.2108e-04 - val_mae: 0.0078 Epoch 194/200
225/225 □□□□□□□□□□□□□□□□□□□□□□□□ 1s 4ms/step - loss: 8.8202e-05 - mae:
0.0068 - val_loss: 1.1851e-04 - val_mae: 0.0077 Epoch 195/200
225/225 □□□□□□□□□□□□□□□□□□□□□□□□ 1s 4ms/step - loss: 8.7254e-05 - mae:
0.0068 - val_loss: 1.1817e-04 - val_mae: 0.0076 Epoch 196/200
225/225 □□□□□□□□□□□□□□□□□□□□□□□□ 1s 4ms/step - loss: 8.6236e-05 - mae:
0.0067 - val_loss: 1.1786e-04 - val_mae: 0.0077 Epoch 197/200
225/225 □□□□□□□□□□□□□□□□□□□□□□□□ 1s 4ms/step - loss: 8.9760e-05 - mae:
0.0068 - val_loss: 1.1805e-04 - val_mae: 0.0077 Epoch 198/200
225/225 □□□□□□□□□□□□□□□□□□□□□□□□ 1s 4ms/step - loss: 8.8281e-05 - mae:
0.0068 - val_loss: 1.1641e-04 - val_mae: 0.0076 Epoch 199/200
225/225 □□□□□□□□□□□□□□□□□□□□□□□□ 1s 4ms/step - loss: 8.4876e-05 - mae:
0.0067 - val_loss: 1.1839e-04 - val_mae: 0.0077 Epoch 200/200
225/225 □□□□□□□□□□□□□□□□□□□□□□□□ 1s 4ms/step - loss: 8.6105e-05 - mae:
0.0067 - val_loss: 1.2234e-04 - val_mae: 0.0077
```






```
1/1 □□□□□□□□□□□□□□□□□□□□□□□□ 0s 102ms/step

 Model training, evaluation, and submission complete!
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.metrics import (
    mean_squared_error, mean_absolute_error, r2_score,
accuracy_score, f1_score, recall_score, precision_score,
classification_report, confusion_matrix
)
```

```
import tensorflow as tf
import os

# ---------------------------
# 1) Load Data
# ---------------------------train
=
pd.read_csv('/kaggle/input/hull-tactical-market-prediction/train.csv')
test =
pd.read_csv('/kaggle/input/hull-tactical-market-prediction/test.csv')

# Detect target if 'market_forward_excess_returns' in
train.columns:     TARGET =
'market_forward_excess_returns' elif
'forward_returns' in train.columns:     TARGET =
'forward_returns' else:     raise ValueError("No
known target column found.") print(f"Using target:
{TARGET}")

# ---------------------------
# 2) Preprocessing
# ---------------------------
train.fillna(method='ffill', inplace=True)
train.fillna(method='bfill', inplace=True)
test.fillna(method='ffill', inplace=True)
test.fillna(method='bfill', inplace=True)

features = [f for f in train.columns if f not in ['date_id', TARGET]
and f in test.columns]
print(f"Using {len(features)} features.")

scaler = StandardScaler()
X_train = scaler.fit_transform(train[features])
X_test = scaler.transform(test[features])
y_train = train[TARGET].values
y_test = test[TARGET].values if TARGET in test.columns else None

X_train_sub, X_val, y_train_sub, y_val = train_test_split(
    X_train, y_train, test_size=0.2, random_state=42
)

# ---------------------------
# 3) Build Neural Network
# ---------------------------model
= tf.keras.Sequential([
    tf.keras.layers.Dense(256, activation='relu',
input_shape=(X_train.shape[1],)),
tf.keras.layers.Dropout(0.3),
```

```
    tf.keras.layers.Dense(128, activation='relu'),
tf.keras.layers.Dropout(0.2),
    tf.keras.layers.Dense(64, activation='relu'),
tf.keras.layers.Dense(1)
])

model.compile(
    optimizer=tf.keras.optimizers.Adam(0.001),
loss='mean_squared_error',     metrics=['mae']
)

model.summary()

# ---------------------------
# 4) Train Model
# ---------------------------history
= model.fit(
    X_train_sub, y_train_sub,
validation_data=(X_val, y_val),
epochs=200,     batch_size=32,
verbose=1
)

# ---------------------------
# 5) Evaluate Model (Regression Metrics)
# ---------------------------if y_test is not
None:     y_pred_test =
model.predict(X_test).flatten()

    print("=== Regression Metrics ===")
    print("MAE:", mean_absolute_error(y_test, y_pred_test))
print("MSE:", mean_squared_error(y_test, y_pred_test))
print("R2 Score:", r2_score(y_test, y_pred_test))

    # ---------------------------
    # Convert to binary for classification metrics
    # 1 if return > 0 else 0
    y_test_class = (y_test > 0).astype(int)
y_pred_class = (y_pred_test > 0).astype(int)

    print("\n=== Classification Metrics (Positive/Negative) ===")
print("Accuracy:", accuracy_score(y_test_class, y_pred_class))
print("F1 Score:", f1_score(y_test_class, y_pred_class))
print("Recall:", recall_score(y_test_class, y_pred_class))
print("Precision:", precision_score(y_test_class, y_pred_class))
print("\nConfusion Matrix:\n", confusion_matrix(y_test_class,
y_pred_class))
    print("\nClassification Report:\n",
```

```
classification_report(y_test_class, y_pred_class))

# ---------------------------
# 6) Training History Plots
# ---------------------------plt.figure(figsize=(12,5))
plt.plot(history.history['loss'], label='Train Loss')
plt.plot(history.history['val_loss'], label='Val Loss')
plt.xlabel('Epochs'); plt.ylabel('MSE Loss')
plt.title('Training vs Validation Loss') plt.legend()
plt.show()

plt.figure(figsize=(12,5))
plt.plot(history.history['mae'], label='Train MAE')
plt.plot(history.history['val_mae'], label='Val MAE')
plt.xlabel('Epochs'); plt.ylabel('MAE')
plt.title('Training vs Validation MAE') plt.legend()
plt.show()

# ---------------------------
# 7) Save Model
model_save_path = "hull_market_model1.h5"
model.save(model_save_path)
print(f" Model saved at {model_save_path}")


# ---------------------------
# 8) Predictions & Submission
# ---------------------------
test_predictions = model.predict(X_test).flatten()
submission = pd.DataFrame({      'date_id':
test['date_id'],
    'predicted': test_predictions
})
submission.to_csv('submission.csv', index=False)
print(" Submission CSV saved!")
Using target: market_forward_excess_returns
Using 94 features.

Model: "sequential_1"
□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□
□□□□□□

□ Layer (type)                    □ Output Shape              □
Param # □

□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□
□□□□□□
```

```
| dense_4 (Dense)                      | (None, 256)                 |
24,320 |
├──────────────────────────────────────┼─────────────────────────────┼──────────
──────┤
| dropout_2 (Dropout)                  | (None, 256)                 |
0 |
├──────────────────────────────────────┼─────────────────────────────┼──────────
──────┤
| dense_5 (Dense)                      | (None, 128)                 |
32,896 |
├──────────────────────────────────────┼─────────────────────────────┼──────────
──────┤
| dropout_3 (Dropout)                  | (None, 128)                 |
0 |
├──────────────────────────────────────┼─────────────────────────────┼──────────
──────┤
| dense_6 (Dense)                      | (None, 64)                  |
8,256 |
├──────────────────────────────────────┼─────────────────────────────┼──────────
──────┤
| dense_7 (Dense)                      | (None, 1)                   |
65 |
└──────────────────────────────────────┴─────────────────────────────┴──────────
──────┘

 Total params: 65,537 (256.00 KB)

 Trainable params: 65,537 (256.00 KB)  Non-

trainable params: 0 (0.00 B)

Epoch 1/200
225/225 □□□□□□□□□□□□□□□□□□□□□□□□ 4s 6ms/step - loss: 0.0372 - mae: 0.1235
- val_loss: 4.3721e-04 - val_mae: 0.0159
Epoch 2/200
225/225 □□□□□□□□□□□□□□□□□□□□□□□□ 1s 4ms/step - loss: 0.0015 - mae: 0.0282
- val_loss: 2.0015e-04 - val_mae: 0.0106
Epoch 3/200 225/225 □□□□□□□□□□□□□□□□□□□□□□□□ 1s 4ms/step - loss:
6.0354e-04 - mae:
0.0176 - val_loss: 1.3571e-04 - val_mae: 0.0086 Epoch 4/200
225/225 □□□□□□□□□□□□□□□□□□□□□□□□ 1s 4ms/step - loss: 3.7225e-04 - mae:
0.0139 - val_loss: 1.2819e-04 - val_mae: 0.0083 Epoch 5/200
225/225 □□□□□□□□□□□□□□□□□□□□□□□□ 1s 4ms/step - loss: 2.7903e-04 - mae:
0.0117 - val_loss: 1.1949e-04 - val_mae: 0.0079 Epoch 6/200
225/225 □□□□□□□□□□□□□□□□□□□□□□□□ 1s 4ms/step - loss: 2.0649e-04 - mae:
0.0104 - val_loss: 1.1979e-04 - val_mae: 0.0079
Epoch 7/200
```

```
225/225 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - loss: 1.9073e-04 - mae:
0.0098 - val_loss: 1.1626e-04 - val_mae: 0.0078
Epoch 8/200
225/225 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - loss: 1.7016e-04 - mae:
0.0092 - val_loss: 1.1609e-04 - val_mae: 0.0077
Epoch 9/200
225/225 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - loss: 1.5355e-04 - mae:
0.0089 - val_loss: 1.1205e-04 - val_mae: 0.0076
Epoch 10/200
225/225 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - loss: 1.4157e-04 - mae:
0.0086 - val_loss: 1.1183e-04 - val_mae: 0.0075
Epoch 11/200
225/225 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - loss: 1.3669e-04 - mae:
0.0084 - val_loss: 1.1191e-04 - val_mae: 0.0076
Epoch 12/200
225/225 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - loss: 1.3215e-04 - mae:
0.0084 - val_loss: 1.2496e-04 - val_mae: 0.0082
Epoch 13/200
225/225 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - loss: 1.3249e-04 - mae:
0.0083 - val_loss: 1.1566e-04 - val_mae: 0.0077
Epoch 14/200
225/225 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - loss: 1.3213e-04 - mae:
0.0082 - val_loss: 1.1209e-04 - val_mae: 0.0075
Epoch 15/200
225/225 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - loss: 1.2069e-04 - mae:
0.0079 - val_loss: 1.1289e-04 - val_mae: 0.0075
Epoch 16/200
225/225 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - loss: 1.2551e-04 - mae:
0.0081 - val_loss: 1.3112e-04 - val_mae: 0.0085
Epoch 17/200
225/225 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - loss: 1.2837e-04 - mae:
0.0082 - val_loss: 1.2517e-04 - val_mae: 0.0081
Epoch 18/200
225/225 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - loss: 1.2622e-04 - mae:
0.0081 - val_loss: 1.1471e-04 - val_mae: 0.0076
Epoch 19/200
225/225 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - loss: 1.2044e-04 - mae:
```

```
0.0079 - val_loss: 1.1380e-04 - val_mae: 0.0077
Epoch 20/200

225/225 □□□□□□□□□□□□□□□□□□□□□□□□ 1s 4ms/step - loss: 1.2009e-04 - mae:
0.0079 - val_loss: 1.1111e-04 - val_mae: 0.0075
Epoch 21/200

225/225 □□□□□□□□□□□□□□□□□□□□□□□□ 1s 4ms/step - loss: 1.2135e-04 - mae:
0.0080 - val_loss: 1.1372e-04 - val_mae: 0.0077
Epoch 22/200

225/225 □□□□□□□□□□□□□□□□□□□□□□□□ 1s 4ms/step - loss: 1.1314e-04 - mae:
0.0076 - val_loss: 1.1238e-04 - val_mae: 0.0075
Epoch 23/200

225/225 □□□□□□□□□□□□□□□□□□□□□□□□ 1s 4ms/step - loss: 1.2061e-04 - mae:
```

```
0.0078 - val_loss: 1.1279e-04 - val_mae: 0.0075 Epoch 24/200
225/225 □□□□□□□□□□□□□□□□□□□□ 1s 4ms/step - loss: 1.2060e-04 - mae:
0.0079 - val_loss: 1.1327e-04 - val_mae: 0.0076 Epoch 25/200
225/225 □□□□□□□□□□□□□□□□□□□□ 1s 4ms/step - loss: 1.1869e-04 - mae:
0.0079 - val_loss: 1.1713e-04 - val_mae: 0.0077 Epoch 26/200
225/225 □□□□□□□□□□□□□□□□□□□□ 1s 4ms/step - loss: 1.1641e-04 - mae:
0.0077 - val_loss: 1.1723e-04 - val_mae: 0.0079 Epoch 27/200
225/225 □□□□□□□□□□□□□□□□□□□□ 1s 4ms/step - loss: 1.1118e-04 - mae:
0.0075 - val_loss: 1.1950e-04 - val_mae: 0.0080 Epoch 28/200
225/225 □□□□□□□□□□□□□□□□□□□□ 1s 4ms/step - loss: 1.1504e-04 - mae:
0.0076 - val_loss: 1.1142e-04 - val_mae: 0.0075 Epoch 29/200
225/225 □□□□□□□□□□□□□□□□□□□□ 1s 4ms/step - loss: 1.1296e-04 - mae:
0.0077 - val_loss: 1.1356e-04 - val_mae: 0.0076 Epoch 30/200
225/225 □□□□□□□□□□□□□□□□□□□□ 1s 4ms/step - loss: 1.1573e-04 - mae:
0.0077 - val_loss: 1.1209e-04 - val_mae: 0.0075 Epoch 31/200
225/225 □□□□□□□□□□□□□□□□□□□□ 1s 4ms/step - loss: 1.2007e-04 - mae:
0.0078 - val_loss: 1.1433e-04 - val_mae: 0.0076 Epoch 32/200
225/225 □□□□□□□□□□□□□□□□□□□□ 1s 4ms/step - loss: 1.0817e-04 - mae:
0.0074 - val_loss: 1.1882e-04 - val_mae: 0.0078 Epoch 33/200
225/225 □□□□□□□□□□□□□□□□□□□□ 1s 4ms/step - loss: 1.0803e-04 - mae:
0.0074 - val_loss: 1.1465e-04 - val_mae: 0.0076 Epoch 34/200
225/225 □□□□□□□□□□□□□□□□□□□□ 1s 4ms/step - loss: 1.1312e-04 - mae:
0.0077 - val_loss: 1.1168e-04 - val_mae: 0.0075 Epoch 35/200
225/225 □□□□□□□□□□□□□□□□□□□□ 1s 4ms/step - loss: 1.1044e-04 - mae:
0.0075 - val_loss: 1.1159e-04 - val_mae: 0.0075 Epoch 36/200
225/225 □□□□□□□□□□□□□□□□□□□□ 1s 4ms/step - loss: 1.1640e-04 - mae:
0.0077 - val_loss: 1.1295e-04 - val_mae: 0.0075 Epoch 37/200
225/225 □□□□□□□□□□□□□□□□□□□□ 1s 4ms/step - loss: 1.1352e-04 - mae:
0.0076 - val_loss: 1.1227e-04 - val_mae: 0.0075 Epoch 38/200
225/225 □□□□□□□□□□□□□□□□□□□□ 1s 4ms/step - loss: 1.0788e-04 - mae:
0.0074 - val_loss: 1.2367e-04 - val_mae: 0.0083 Epoch 39/200
225/225 □□□□□□□□□□□□□□□□□□□□ 1s 4ms/step - loss: 1.0962e-04 - mae:
0.0075 - val_loss: 1.1273e-04 - val_mae: 0.0076
Epoch 40/200 225/225 □□□□□□□□□□□□□□□□□□□□ 1s 4ms/step - loss:
1.1532e-04 - mae:
0.0077 - val_loss: 1.1294e-04 - val_mae: 0.0076 Epoch 41/200
225/225 □□□□□□□□□□□□□□□□□□□□ 1s 4ms/step - loss: 1.1161e-04 - mae:
0.0075 - val_loss: 1.1148e-04 - val_mae: 0.0075 Epoch 42/200
225/225 □□□□□□□□□□□□□□□□□□□□ 1s 4ms/step - loss: 1.1212e-04 - mae:
0.0076 - val_loss: 1.1429e-04 - val_mae: 0.0077 Epoch 43/200
225/225 □□□□□□□□□□□□□□□□□□□□ 1s 4ms/step - loss: 1.1969e-04 - mae:
0.0078 - val_loss: 1.1563e-04 - val_mae: 0.0077 Epoch 44/200
225/225 □□□□□□□□□□□□□□□□□□□□ 1s 4ms/step - loss: 1.0885e-04 - mae:
0.0075 - val_loss: 1.1200e-04 - val_mae: 0.0075 Epoch 45/200
225/225 □□□□□□□□□□□□□□□□□□□□ 1s 4ms/step - loss: 1.1089e-04 - mae:
0.0075 - val_loss: 1.1218e-04 - val_mae: 0.0075 Epoch 46/200
225/225 □□□□□□□□□□□□□□□□□□□□ 1s 4ms/step - loss: 1.1245e-04 - mae:
```

0.0075 - val_loss: 1.1131e-04 - val_mae: 0.0075 Epoch 47/200
225/225 □□□□□□□□□□□□□□□□□□□□□□□□ 1s 4ms/step - loss: 1.1196e-04 - mae:
0.0076 - val_loss: 1.1238e-04 - val_mae: 0.0075 Epoch 48/200
225/225 □□□□□□□□□□□□□□□□□□□□□□□□ 1s 4ms/step - loss: 1.0816e-04 - mae:
0.0075 - val_loss: 1.1188e-04 - val_mae: 0.0075 Epoch 49/200
225/225 □□□□□□□□□□□□□□□□□□□□□□□□ 1s 4ms/step - loss: 1.1151e-04 - mae:
0.0075 - val_loss: 1.1407e-04 - val_mae: 0.0075 Epoch 50/200
225/225 □□□□□□□□□□□□□□□□□□□□□□□□ 1s 4ms/step - loss: 1.1338e-04 - mae:
0.0076 - val_loss: 1.1258e-04 - val_mae: 0.0075 Epoch 51/200
225/225 □□□□□□□□□□□□□□□□□□□□□□□□ 1s 4ms/step - loss: 1.1221e-04 - mae:
0.0075 - val_loss: 1.1200e-04 - val_mae: 0.0075 Epoch 52/200
225/225 □□□□□□□□□□□□□□□□□□□□□□□□ 1s 4ms/step - loss: 1.0952e-04 - mae:
0.0074 - val_loss: 1.1166e-04 - val_mae: 0.0075 Epoch 53/200
225/225 □□□□□□□□□□□□□□□□□□□□□□□□ 1s 4ms/step - loss: 1.0930e-04 - mae:
0.0075 - val_loss: 1.1153e-04 - val_mae: 0.0075 Epoch 54/200
225/225 □□□□□□□□□□□□□□□□□□□□□□□□ 1s 4ms/step - loss: 1.0849e-04 - mae:
0.0075 - val_loss: 1.1409e-04 - val_mae: 0.0076 Epoch 55/200
225/225 □□□□□□□□□□□□□□□□□□□□□□□□ 1s 4ms/step - loss: 1.1119e-04 - mae:
0.0075 - val_loss: 1.1266e-04 - val_mae: 0.0075
Epoch 56/200

```
225/225 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - loss: 1.1150e-04 - mae:
0.0075 - val_loss: 1.1108e-04 - val_mae: 0.0075
Epoch 57/200

225/225 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - loss: 1.0947e-04 - mae:
0.0075 - val_loss: 1.1138e-04 - val_mae: 0.0075
Epoch 58/200

225/225 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - loss: 1.1011e-04 - mae:
0.0075 - val_loss: 1.1091e-04 - val_mae: 0.0075
Epoch 59/200

225/225 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - loss: 1.0903e-04 - mae:
0.0075 - val_loss: 1.1249e-04 - val_mae: 0.0075
Epoch 60/200

225/225 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - loss: 1.1311e-04 - mae:
0.0076 - val_loss: 1.1191e-04 - val_mae: 0.0075
Epoch 61/200

225/225 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - loss: 1.0850e-04 - mae:
0.0074 - val_loss: 1.1449e-04 - val_mae: 0.0077
Epoch 62/200

225/225 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - loss: 1.0416e-04 - mae:
0.0073 - val_loss: 1.1229e-04 - val_mae: 0.0075
Epoch 63/200

225/225 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - loss: 1.1011e-04 - mae:
0.0075 - val_loss: 1.1377e-04 - val_mae: 0.0076
Epoch 64/200

225/225 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - loss: 1.0696e-04 - mae:
0.0074 - val_loss: 1.1241e-04 - val_mae: 0.0075
Epoch 65/200

225/225 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - loss: 1.1195e-04 - mae:
0.0075 - val_loss: 1.1790e-04 - val_mae: 0.0077
Epoch 66/200

225/225 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - loss: 1.1115e-04 - mae:
0.0076 - val_loss: 1.1390e-04 - val_mae: 0.0076
Epoch 67/200

225/225 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - loss: 1.0696e-04 - mae:
0.0074 - val_loss: 1.1437e-04 - val_mae: 0.0076
Epoch 68/200

225/225 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - loss: 1.0485e-04 - mae:
```

```
0.0073 - val_loss: 1.1583e-04 - val_mae: 0.0076
Epoch 69/200

225/225 □□□□□□□□□□□□□□□□□□□□□□□□ 1s 4ms/step - loss: 1.0555e-04 - mae:
0.0074 - val_loss: 1.1880e-04 - val_mae: 0.0076
Epoch 70/200

225/225 □□□□□□□□□□□□□□□□□□□□□□□□ 1s 4ms/step - loss: 1.0961e-04 - mae:
0.0075 - val_loss: 1.1609e-04 - val_mae: 0.0075
Epoch 71/200

225/225 □□□□□□□□□□□□□□□□□□□□□□□□ 1s 4ms/step - loss: 1.2863e-04 - mae:
0.0074 - val_loss: 1.1687e-04 - val_mae: 0.0076
Epoch 72/200

225/225 □□□□□□□□□□□□□□□□□□□□□□□□ 1s 4ms/step - loss: 1.0885e-04 - mae:
```

```
0.0074 - val_loss: 1.1473e-04 - val_mae: 0.0076 Epoch 73/200
225/225 □□□□□□□□□□□□□□□□□□□□ 1s 4ms/step - loss: 1.0442e-04 - mae:
0.0073 - val_loss: 1.1149e-04 - val_mae: 0.0075 Epoch 74/200
225/225 □□□□□□□□□□□□□□□□□□□□ 1s 4ms/step - loss: 1.0527e-04 - mae:
0.0073 - val_loss: 1.1464e-04 - val_mae: 0.0077 Epoch 75/200
225/225 □□□□□□□□□□□□□□□□□□□□ 1s 4ms/step - loss: 1.0689e-04 - mae:
0.0074 - val_loss: 1.2213e-04 - val_mae: 0.0081 Epoch 76/200
225/225 □□□□□□□□□□□□□□□□□□□□ 1s 4ms/step - loss: 1.0736e-04 - mae:
0.0074 - val_loss: 1.1193e-04 - val_mae: 0.0075 Epoch 77/200
225/225 □□□□□□□□□□□□□□□□□□□□ 1s 4ms/step - loss: 1.0147e-04 - mae:
0.0072 - val_loss: 1.1343e-04 - val_mae: 0.0075 Epoch 78/200
225/225 □□□□□□□□□□□□□□□□□□□□ 1s 4ms/step - loss: 1.0592e-04 - mae:
0.0073 - val_loss: 1.1256e-04 - val_mae: 0.0076 Epoch 79/200
225/225 □□□□□□□□□□□□□□□□□□□□ 1s 4ms/step - loss: 1.0290e-04 - mae:
0.0073 - val_loss: 1.1624e-04 - val_mae: 0.0076 Epoch 80/200
225/225 □□□□□□□□□□□□□□□□□□□□ 1s 4ms/step - loss: 1.0623e-04 - mae:
0.0074 - val_loss: 1.1106e-04 - val_mae: 0.0075 Epoch 81/200
225/225 □□□□□□□□□□□□□□□□□□□□ 1s 4ms/step - loss: 1.0768e-04 - mae:
0.0074 - val_loss: 1.1247e-04 - val_mae: 0.0075 Epoch 82/200
225/225 □□□□□□□□□□□□□□□□□□□□ 1s 4ms/step - loss: 9.7899e-05 - mae:
0.0071 - val_loss: 1.1374e-04 - val_mae: 0.0077 Epoch 83/200
225/225 □□□□□□□□□□□□□□□□□□□□ 1s 4ms/step - loss: 1.1122e-04 - mae:
0.0073 - val_loss: 1.1471e-04 - val_mae: 0.0075 Epoch 84/200
225/225 □□□□□□□□□□□□□□□□□□□□ 1s 4ms/step - loss: 1.0825e-04 - mae:
0.0074 - val_loss: 1.1614e-04 - val_mae: 0.0077 Epoch 85/200
225/225 □□□□□□□□□□□□□□□□□□□□ 1s 4ms/step - loss: 1.0593e-04 - mae:
0.0074 - val_loss: 1.1715e-04 - val_mae: 0.0076 Epoch 86/200
225/225 □□□□□□□□□□□□□□□□□□□□ 1s 4ms/step - loss: 1.0387e-04 - mae:
0.0073 - val_loss: 1.1697e-04 - val_mae: 0.0076 Epoch 87/200
225/225 □□□□□□□□□□□□□□□□□□□□ 1s 4ms/step - loss: 1.0778e-04 - mae:
0.0074 - val_loss: 1.1624e-04 - val_mae: 0.0076 Epoch 88/200
225/225 □□□□□□□□□□□□□□□□□□□□ 1s 4ms/step - loss: 1.0029e-04 - mae:
0.0072 - val_loss: 1.1105e-04 - val_mae: 0.0075
Epoch 89/200 225/225 □□□□□□□□□□□□□□□□□□□□ 1s 4ms/step - loss:
1.0034e-04 - mae:
0.0072 - val_loss: 1.1296e-04 - val_mae: 0.0076 Epoch 90/200
225/225 □□□□□□□□□□□□□□□□□□□□ 1s 4ms/step - loss: 1.0118e-04 - mae:
0.0072 - val_loss: 1.2169e-04 - val_mae: 0.0077 Epoch 91/200
225/225 □□□□□□□□□□□□□□□□□□□□ 1s 5ms/step - loss: 9.9113e-05 - mae:
0.0072 - val_loss: 1.1471e-04 - val_mae: 0.0076 Epoch 92/200
225/225 □□□□□□□□□□□□□□□□□□□□ 1s 5ms/step - loss: 9.9102e-05 - mae:
0.0071 - val_loss: 1.1792e-04 - val_mae: 0.0077 Epoch 93/200
225/225 □□□□□□□□□□□□□□□□□□□□ 1s 4ms/step - loss: 9.7489e-05 - mae:
0.0071 - val_loss: 1.2165e-04 - val_mae: 0.0079 Epoch 94/200
225/225 □□□□□□□□□□□□□□□□□□□□ 1s 5ms/step - loss: 1.0029e-04 - mae:
0.0071 - val_loss: 1.1320e-04 - val_mae: 0.0076 Epoch 95/200
225/225 □□□□□□□□□□□□□□□□□□□□ 1s 4ms/step - loss: 9.9275e-05 - mae:
```

0.0072 - val_loss: 1.1229e-04 - val_mae: 0.0075 Epoch 96/200
225/225 □□□□□□□□□□□□□□□□□□□□□□□□ 1s 4ms/step - loss: 9.7089e-05 - mae:
0.0071 - val_loss: 1.1287e-04 - val_mae: 0.0075 Epoch 97/200
225/225 □□□□□□□□□□□□□□□□□□□□□□□□ 1s 4ms/step - loss: 9.8417e-05 - mae:
0.0071 - val_loss: 1.1943e-04 - val_mae: 0.0078 Epoch 98/200
225/225 □□□□□□□□□□□□□□□□□□□□□□□□ 1s 4ms/step - loss: 9.7072e-05 - mae:
0.0072 - val_loss: 1.1194e-04 - val_mae: 0.0075 Epoch 99/200
225/225 □□□□□□□□□□□□□□□□□□□□□□□□ 1s 4ms/step - loss: 9.3162e-05 - mae:
0.0070 - val_loss: 1.1299e-04 - val_mae: 0.0075 Epoch 100/200
225/225 □□□□□□□□□□□□□□□□□□□□□□□□ 1s 4ms/step - loss: 9.6573e-05 - mae:
0.0071 - val_loss: 1.1323e-04 - val_mae: 0.0075 Epoch 101/200
225/225 □□□□□□□□□□□□□□□□□□□□□□□□ 1s 4ms/step - loss: 1.0380e-04 - mae:
0.0073 - val_loss: 1.1399e-04 - val_mae: 0.0076 Epoch 102/200
225/225 □□□□□□□□□□□□□□□□□□□□□□□□ 1s 4ms/step - loss: 8.9019e-05 - mae:
0.0068 - val_loss: 1.1420e-04 - val_mae: 0.0076 Epoch 103/200
225/225 □□□□□□□□□□□□□□□□□□□□□□□□ 1s 4ms/step - loss: 9.3935e-05 - mae:
0.0070 - val_loss: 1.1739e-04 - val_mae: 0.0077 Epoch 104/200
225/225 □□□□□□□□□□□□□□□□□□□□□□□□ 1s 4ms/step - loss: 9.3538e-05 - mae:
0.0070 - val_loss: 1.1481e-04 - val_mae: 0.0076
Epoch 105/200

```
225/225 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - loss: 9.2153e-05 - mae:
0.0070 - val_loss: 1.1387e-04 - val_mae: 0.0076
Epoch 106/200

225/225 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - loss: 8.8243e-05 - mae:
0.0068 - val_loss: 1.1794e-04 - val_mae: 0.0076
Epoch 107/200

225/225 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - loss: 9.4157e-05 - mae:
0.0070 - val_loss: 1.1431e-04 - val_mae: 0.0075
Epoch 108/200

225/225 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - loss: 9.4142e-05 - mae:
0.0070 - val_loss: 1.1365e-04 - val_mae: 0.0075
Epoch 109/200

225/225 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - loss: 9.0443e-05 - mae:
0.0069 - val_loss: 1.1166e-04 - val_mae: 0.0075
Epoch 110/200

225/225 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - loss: 9.0979e-05 - mae:
0.0069 - val_loss: 1.1525e-04 - val_mae: 0.0077
Epoch 111/200

225/225 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - loss: 8.8835e-05 - mae:
0.0069 - val_loss: 1.1650e-04 - val_mae: 0.0076
Epoch 112/200

225/225 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - loss: 9.2008e-05 - mae:
0.0070 - val_loss: 1.1558e-04 - val_mae: 0.0075
Epoch 113/200

225/225 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - loss: 9.6811e-05 - mae:
0.0068 - val_loss: 1.1310e-04 - val_mae: 0.0075
Epoch 114/200

225/225 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - loss: 8.7330e-05 - mae:
0.0069 - val_loss: 1.1326e-04 - val_mae: 0.0075
Epoch 115/200

225/225 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - loss: 8.4890e-05 - mae:
0.0067 - val_loss: 1.1164e-04 - val_mae: 0.0075
Epoch 116/200

225/225 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - loss: 8.3492e-05 - mae:
0.0067 - val_loss: 1.1587e-04 - val_mae: 0.0078
Epoch 117/200

225/225 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - loss: 7.9499e-05 - mae:
```

```
0.0065 - val_loss: 1.1654e-04 - val_mae: 0.0076
Epoch 118/200

225/225 □□□□□□□□□□□□□□□□□□□□□□□□ 1s 4ms/step - loss: 8.3927e-05 - mae:
0.0067 - val_loss: 1.0992e-04 - val_mae: 0.0075
Epoch 119/200

225/225 □□□□□□□□□□□□□□□□□□□□□□□□ 1s 4ms/step - loss: 8.9668e-05 - mae:
0.0069 - val_loss: 1.1913e-04 - val_mae: 0.0077
Epoch 120/200

225/225 □□□□□□□□□□□□□□□□□□□□□□□□ 1s 4ms/step - loss: 8.3137e-05 - mae:
0.0066 - val_loss: 1.1948e-04 - val_mae: 0.0076
Epoch 121/200

225/225 □□□□□□□□□□□□□□□□□□□□□□□□ 1s 4ms/step - loss: 8.4227e-05 - mae:
```

```
0.0067 - val_loss: 1.2045e-04 - val_mae: 0.0079 Epoch 122/200
225/225 □□□□□□□□□□□□□□□□□□□□ 1s 4ms/step - loss: 8.6991e-05 - mae:
0.0067 - val_loss: 1.1777e-04 - val_mae: 0.0076 Epoch 123/200
225/225 □□□□□□□□□□□□□□□□□□□□ 1s 4ms/step - loss: 8.6405e-05 - mae:
0.0068 - val_loss: 1.1429e-04 - val_mae: 0.0076 Epoch 124/200
225/225 □□□□□□□□□□□□□□□□□□□□ 1s 4ms/step - loss: 8.2176e-05 - mae:
0.0066 - val_loss: 1.2662e-04 - val_mae: 0.0082 Epoch 125/200
225/225 □□□□□□□□□□□□□□□□□□□□ 1s 4ms/step - loss: 8.9873e-05 - mae:
0.0068 - val_loss: 1.1269e-04 - val_mae: 0.0075 Epoch 126/200
225/225 □□□□□□□□□□□□□□□□□□□□ 1s 4ms/step - loss: 8.2066e-05 - mae:
0.0066 - val_loss: 1.1306e-04 - val_mae: 0.0075 Epoch 127/200
225/225 □□□□□□□□□□□□□□□□□□□□ 1s 4ms/step - loss: 8.0488e-05 - mae:
0.0066 - val_loss: 1.2048e-04 - val_mae: 0.0076 Epoch 128/200
225/225 □□□□□□□□□□□□□□□□□□□□ 1s 4ms/step - loss: 8.0543e-05 - mae:
0.0066 - val_loss: 1.1593e-04 - val_mae: 0.0076 Epoch 129/200
225/225 □□□□□□□□□□□□□□□□□□□□ 1s 4ms/step - loss: 7.6532e-05 - mae:
0.0064 - val_loss: 1.1716e-04 - val_mae: 0.0076 Epoch 130/200
225/225 □□□□□□□□□□□□□□□□□□□□ 1s 4ms/step - loss: 8.1631e-05 - mae:
0.0067 - val_loss: 1.1376e-04 - val_mae: 0.0075 Epoch 131/200
225/225 □□□□□□□□□□□□□□□□□□□□ 1s 4ms/step - loss: 8.0276e-05 - mae:
0.0066 - val_loss: 1.1898e-04 - val_mae: 0.0077 Epoch 132/200
225/225 □□□□□□□□□□□□□□□□□□□□ 1s 4ms/step - loss: 7.9049e-05 - mae:
0.0064 - val_loss: 1.1840e-04 - val_mae: 0.0077 Epoch 133/200
225/225 □□□□□□□□□□□□□□□□□□□□ 1s 4ms/step - loss: 7.8297e-05 - mae:
0.0065 - val_loss: 1.2247e-04 - val_mae: 0.0077 Epoch 134/200
225/225 □□□□□□□□□□□□□□□□□□□□ 1s 4ms/step - loss: 7.8689e-05 - mae:
0.0066 - val_loss: 1.1836e-04 - val_mae: 0.0076 Epoch 135/200
225/225 □□□□□□□□□□□□□□□□□□□□ 1s 4ms/step - loss: 7.6480e-05 - mae:
0.0064 - val_loss: 1.1981e-04 - val_mae: 0.0077 Epoch 136/200
225/225 □□□□□□□□□□□□□□□□□□□□ 1s 4ms/step - loss: 8.2654e-05 - mae:
0.0066 - val_loss: 1.1905e-04 - val_mae: 0.0077 Epoch 137/200
225/225 □□□□□□□□□□□□□□□□□□□□ 1s 4ms/step - loss: 7.6037e-05 - mae:
0.0065 - val_loss: 1.1808e-04 - val_mae: 0.0076
Epoch 138/200 225/225 □□□□□□□□□□□□□□□□□□□□ 1s 4ms/step - loss:
7.5471e-05 - mae:
0.0064 - val_loss: 1.1971e-04 - val_mae: 0.0077 Epoch 139/200
225/225 □□□□□□□□□□□□□□□□□□□□ 1s 4ms/step - loss: 7.6336e-05 - mae:
0.0064 - val_loss: 1.2470e-04 - val_mae: 0.0079 Epoch 140/200
225/225 □□□□□□□□□□□□□□□□□□□□ 1s 4ms/step - loss: 7.8074e-05 - mae:
0.0065 - val_loss: 1.1873e-04 - val_mae: 0.0077 Epoch 141/200
225/225 □□□□□□□□□□□□□□□□□□□□ 1s 4ms/step - loss: 7.6469e-05 - mae:
0.0064 - val_loss: 1.1843e-04 - val_mae: 0.0077 Epoch 142/200
225/225 □□□□□□□□□□□□□□□□□□□□ 1s 4ms/step - loss: 7.5680e-05 - mae:
0.0064 - val_loss: 1.1768e-04 - val_mae: 0.0077 Epoch 143/200
225/225 □□□□□□□□□□□□□□□□□□□□ 1s 4ms/step - loss: 7.6916e-05 - mae:
0.0064 - val_loss: 1.2248e-04 - val_mae: 0.0077 Epoch 144/200
225/225 □□□□□□□□□□□□□□□□□□□□ 1s 4ms/step - loss: 7.5506e-05 - mae:
```

0.0064 - val_loss: 1.2342e-04 - val_mae: 0.0077 Epoch 145/200
225/225 □□□□□□□□□□□□□□□□□□□□□□□□ 1s 4ms/step - loss: 6.7915e-05 - mae:
0.0061 - val_loss: 1.1793e-04 - val_mae: 0.0076 Epoch 146/200
225/225 □□□□□□□□□□□□□□□□□□□□□□□□ 1s 4ms/step - loss: 7.0738e-05 - mae:
0.0062 - val_loss: 1.2261e-04 - val_mae: 0.0077 Epoch 147/200
225/225 □□□□□□□□□□□□□□□□□□□□□□□□ 1s 4ms/step - loss: 7.1501e-05 - mae:
0.0063 - val_loss: 1.1904e-04 - val_mae: 0.0076 Epoch 148/200
225/225 □□□□□□□□□□□□□□□□□□□□□□□□ 1s 4ms/step - loss: 7.0228e-05 - mae:
0.0062 - val_loss: 1.1517e-04 - val_mae: 0.0076 Epoch 149/200
225/225 □□□□□□□□□□□□□□□□□□□□□□□□ 1s 4ms/step - loss: 7.0977e-05 - mae:
0.0061 - val_loss: 1.1650e-04 - val_mae: 0.0076 Epoch 150/200
225/225 □□□□□□□□□□□□□□□□□□□□□□□□ 1s 4ms/step - loss: 7.3411e-05 - mae:
0.0063 - val_loss: 1.1491e-04 - val_mae: 0.0076 Epoch 151/200
225/225 □□□□□□□□□□□□□□□□□□□□□□□□ 1s 5ms/step - loss: 6.7400e-05 - mae:
0.0061 - val_loss: 1.1639e-04 - val_mae: 0.0076 Epoch 152/200
225/225 □□□□□□□□□□□□□□□□□□□□□□□□ 1s 5ms/step - loss: 6.8636e-05 - mae:
0.0062 - val_loss: 1.1721e-04 - val_mae: 0.0076 Epoch 153/200
225/225 □□□□□□□□□□□□□□□□□□□□□□□□ 1s 4ms/step - loss: 6.8302e-05 - mae:
0.0062 - val_loss: 1.1967e-04 - val_mae: 0.0077
Epoch 154/200

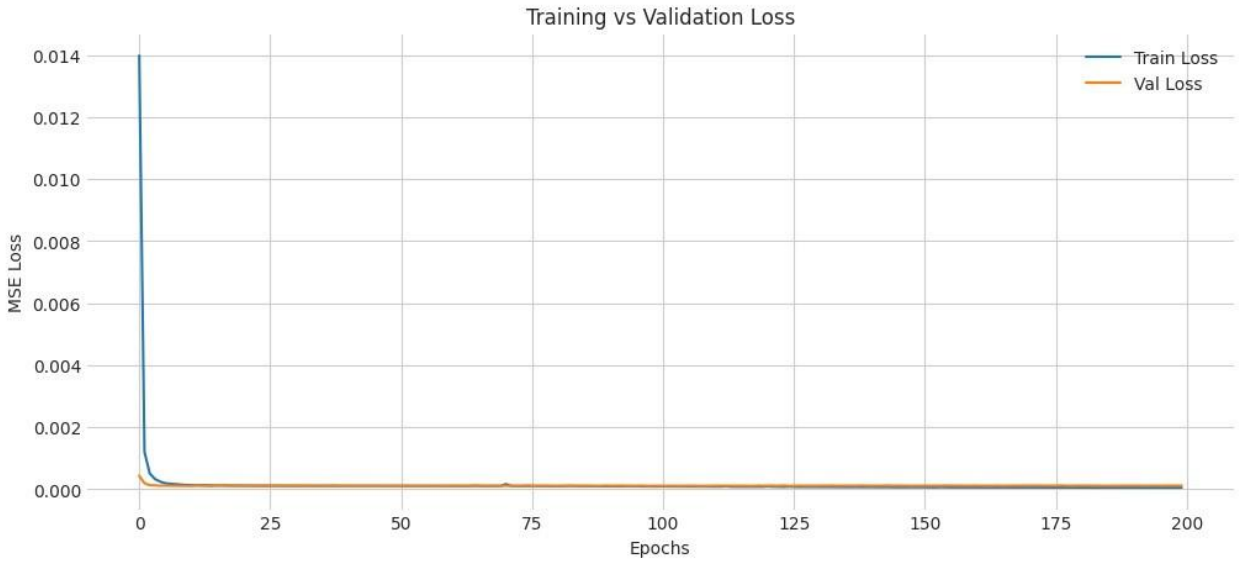
```
225/225 □□□□□□□□□□□□□□□□□□□□□□□□ 1s 4ms/step - loss: 6.9665e-05 - mae:
0.0062 - val_loss: 1.1857e-04 - val_mae: 0.0076
Epoch 155/200

225/225 □□□□□□□□□□□□□□□□□□□□□□□□ 1s 4ms/step - loss: 7.1019e-05 - mae:
0.0062 - val_loss: 1.2281e-04 - val_mae: 0.0078
Epoch 156/200

225/225 □□□□□□□□□□□□□□□□□□□□□□□□ 1s 5ms/step - loss: 6.6814e-05 - mae:
0.0060 - val_loss: 1.2005e-04 - val_mae: 0.0077
Epoch 157/200

225/225 □□□□□□□□□□□□□□□□□□□□□□□□ 1s 4ms/step - loss: 7.0358e-05 - mae:
0.0062 - val_loss: 1.1482e-04 - val_mae: 0.0076
Epoch 158/200

225/225 □□□□□□□□□□□□□□□□□□□□□□□□ 1s 4ms/step - loss: 6.3729e-05 - mae:
0.0059 - val_loss: 1.1892e-04 - val_mae: 0.0077
Epoch 159/200

225/225 □□□□□□□□□□□□□□□□□□□□□□□□ 1s 4ms/step - loss: 6.7843e-05 - mae:
0.0061 - val_loss: 1.1787e-04 - val_mae: 0.0077
Epoch 160/200

225/225 □□□□□□□□□□□□□□□□□□□□□□□□ 1s 4ms/step - loss: 6.4623e-05 - mae:
0.0060 - val_loss: 1.1887e-04 - val_mae: 0.0077
Epoch 161/200

225/225 □□□□□□□□□□□□□□□□□□□□□□□□ 1s 5ms/step - loss: 6.7027e-05 - mae:
0.0061 - val_loss: 1.2047e-04 - val_mae: 0.0077
Epoch 162/200

225/225 □□□□□□□□□□□□□□□□□□□□□□□□ 1s 4ms/step - loss: 6.6695e-05 - mae:
0.0061 - val_loss: 1.1663e-04 - val_mae: 0.0076
Epoch 163/200

225/225 □□□□□□□□□□□□□□□□□□□□□□□□ 1s 4ms/step - loss: 6.9203e-05 - mae:
0.0061 - val_loss: 1.1926e-04 - val_mae: 0.0077
Epoch 164/200

225/225 □□□□□□□□□□□□□□□□□□□□□□□□ 1s 4ms/step - loss: 6.6855e-05 - mae:
0.0061 - val_loss: 1.1961e-04 - val_mae: 0.0077
Epoch 165/200

225/225 □□□□□□□□□□□□□□□□□□□□□□□□ 1s 4ms/step - loss: 6.3104e-05 - mae:
0.0059 - val_loss: 1.2049e-04 - val_mae: 0.0078
Epoch 166/200

225/225 □□□□□□□□□□□□□□□□□□□□□□□□ 1s 4ms/step - loss: 6.5635e-05 - mae:
0.0060 - val_loss: 1.1549e-04 - val_mae: 0.0076
```

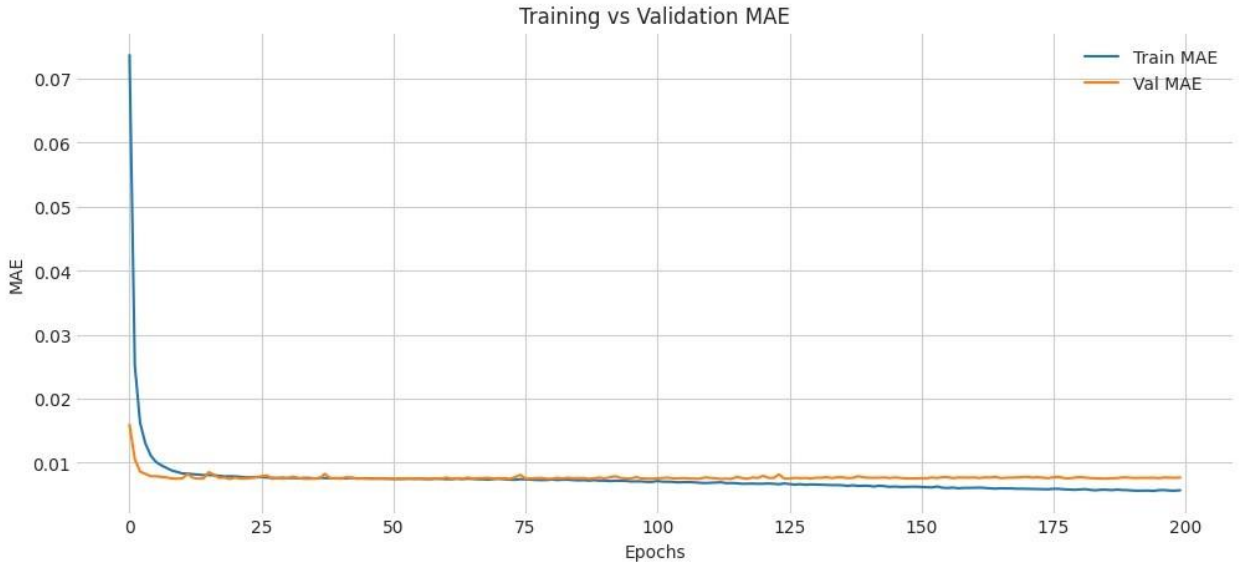
```
Epoch 167/200
225/225 □□□□□□□□□□□□□□□□□□□□□□□□ 1s 4ms/step - loss: 6.4247e-05 - mae:
0.0060 - val_loss: 1.2007e-04 - val_mae: 0.0077
Epoch 168/200
225/225 □□□□□□□□□□□□□□□□□□□□□□□□ 1s 4ms/step - loss: 6.2742e-05 - mae:
0.0059 - val_loss: 1.2091e-04 - val_mae: 0.0077
Epoch 169/200
225/225 □□□□□□□□□□□□□□□□□□□□□□□□ 1s 4ms/step - loss: 6.2704e-05 - mae:
0.0059 - val_loss: 1.2152e-04 - val_mae: 0.0077
Epoch 170/200
225/225 □□□□□□□□□□□□□□□□□□□□□□□□ 1s 4ms/step - loss: 6.2084e-05 - mae:
0.0058 - val_loss: 1.2248e-04 - val_mae: 0.0078
Epoch 171/200 225/225 □□□□□□□□□□□□□□□□□□□□□□□□ 1s 4ms/step - loss:
6.3324e-05 - mae:
0.0059 - val_loss: 1.2576e-04 - val_mae: 0.0078 Epoch 172/200
225/225 □□□□□□□□□□□□□□□□□□□□□□□□ 1s 4ms/step - loss: 6.3157e-05 - mae:
0.0059 - val_loss: 1.2117e-04 - val_mae: 0.0077 Epoch 173/200
225/225 □□□□□□□□□□□□□□□□□□□□□□□□ 1s 4ms/step - loss: 6.1978e-05 - mae:
0.0059 - val_loss: 1.2271e-04 - val_mae: 0.0077 Epoch 174/200
225/225 □□□□□□□□□□□□□□□□□□□□□□□□ 1s 4ms/step - loss: 6.2471e-05 - mae:
0.0059 - val_loss: 1.1970e-04 - val_mae: 0.0077 Epoch 175/200
225/225 □□□□□□□□□□□□□□□□□□□□□□□□ 1s 4ms/step - loss: 6.1266e-05 - mae:
0.0058 - val_loss: 1.1642e-04 - val_mae: 0.0076 Epoch 176/200
225/225 □□□□□□□□□□□□□□□□□□□□□□□□ 1s 4ms/step - loss: 5.9918e-05 - mae:
0.0058 - val_loss: 1.1807e-04 - val_mae: 0.0077 Epoch 177/200
225/225 □□□□□□□□□□□□□□□□□□□□□□□□ 1s 4ms/step - loss: 6.1089e-05 - mae:
0.0058 - val_loss: 1.2605e-04 - val_mae: 0.0078 Epoch 178/200
225/225 □□□□□□□□□□□□□□□□□□□□□□□□ 1s 4ms/step - loss: 6.2217e-05 - mae:
0.0059 - val_loss: 1.1742e-04 - val_mae: 0.0076 Epoch 179/200
225/225 □□□□□□□□□□□□□□□□□□□□□□□□ 1s 4ms/step - loss: 5.9892e-05 - mae:
0.0057 - val_loss: 1.1712e-04 - val_mae: 0.0076 Epoch 180/200
225/225 □□□□□□□□□□□□□□□□□□□□□□□□ 1s 4ms/step - loss: 5.8056e-05 - mae:
0.0057 - val_loss: 1.2141e-04 - val_mae: 0.0077 Epoch 181/200
225/225 □□□□□□□□□□□□□□□□□□□□□□□□ 1s 4ms/step - loss: 5.9381e-05 - mae:
0.0058 - val_loss: 1.2180e-04 - val_mae: 0.0078 Epoch 182/200
225/225 □□□□□□□□□□□□□□□□□□□□□□□□ 1s 4ms/step - loss: 5.9372e-05 - mae:
0.0058 - val_loss: 1.1953e-04 - val_mae: 0.0077 Epoch 183/200
225/225 □□□□□□□□□□□□□□□□□□□□□□□□ 1s 4ms/step - loss: 5.7010e-05 - mae:
0.0057 - val_loss: 1.1945e-04 - val_mae: 0.0076 Epoch 184/200
225/225 □□□□□□□□□□□□□□□□□□□□□□□□ 1s 4ms/step - loss: 5.7537e-05 - mae:
0.0056 - val_loss: 1.1880e-04 - val_mae: 0.0076 Epoch 185/200
225/225 □□□□□□□□□□□□□□□□□□□□□□□□ 1s 4ms/step - loss: 5.8155e-05 - mae:
0.0057 - val_loss: 1.1567e-04 - val_mae: 0.0075 Epoch 186/200
225/225 □□□□□□□□□□□□□□□□□□□□□□□□ 1s 4ms/step - loss: 5.9299e-05 - mae:
```

```
0.0057 - val_loss: 1.1481e-04 - val_mae: 0.0075
Epoch 187/200
```

```
225/225 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - loss: 6.1231e-05 - mae:
0.0057 - val_loss: 1.1868e-04 - val_mae: 0.0076
Epoch 188/200
225/225 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - loss: 6.1428e-05 - mae:
0.0059 - val_loss: 1.1712e-04 - val_mae: 0.0076
Epoch 189/200
225/225 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - loss: 5.7542e-05 - mae:
0.0057 - val_loss: 1.2088e-04 - val_mae: 0.0077
Epoch 190/200
225/225 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - loss: 5.9891e-05 - mae:
0.0057 - val_loss: 1.1793e-04 - val_mae: 0.0077
Epoch 191/200
225/225 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - loss: 5.6504e-05 - mae:
0.0056 - val_loss: 1.1997e-04 - val_mae: 0.0076
Epoch 192/200
225/225 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - loss: 5.4927e-05 - mae:
0.0056 - val_loss: 1.1709e-04 - val_mae: 0.0076
Epoch 193/200
225/225 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - loss: 5.5526e-05 - mae:
0.0056 - val_loss: 1.1664e-04 - val_mae: 0.0076
Epoch 194/200
225/225 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - loss: 5.4343e-05 - mae:
0.0056 - val_loss: 1.1849e-04 - val_mae: 0.0077
Epoch 195/200
225/225 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - loss: 5.3445e-05 - mae:
0.0055 - val_loss: 1.1943e-04 - val_mae: 0.0077
Epoch 196/200
225/225 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - loss: 5.9832e-05 - mae:
0.0058 - val_loss: 1.1578e-04 - val_mae: 0.0076
Epoch 197/200
225/225 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - loss: 6.0076e-05 - mae:
0.0058 - val_loss: 1.1911e-04 - val_mae: 0.0077
Epoch 198/200
225/225 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - loss: 5.4801e-05 - mae:
0.0056 - val_loss: 1.2011e-04 - val_mae: 0.0077
Epoch 199/200
```

```
225/225 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - loss: 5.7081e-05 - mae:
0.0057 - val_loss: 1.1776e-04 - val_mae: 0.0077
Epoch 200/200
225/225 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - loss: 5.7152e-05 - mae:
0.0057 - val_loss: 1.1833e-04 - val_mae: 0.0077
```






```
 Model saved at hull_market_model1.h5
1/1 □□□□□□□□□□□□□□□□□□□□ 0s 96ms/step
 Submission CSV saved!
```

```
 import os import
pandas as pd import
numpy as np
import matplotlib.pyplot as plt
import seaborn as sns import
tensorflow as tf
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import (
    mean_squared_error, mean_absolute_error, r2_score,
```

```
    accuracy_score, f1_score, recall_score, precision_score,
    roc_auc_score, confusion_matrix, roc_curve
)
import pickle

# ---------------------------
# 1) Load Data
# ---------------------------train
=
pd.read_csv('/kaggle/input/hull-tactical-market-prediction/train.csv')
test =
pd.read_csv('/kaggle/input/hull-tactical-market-prediction/test.csv')

TARGET = 'market_forward_excess_returns' if
'market_forward_excess_returns' in train.columns else
'forward_returns'

# ---------------------------
# 2) Feature Selection
# ---------------------------
features = [col for col in train.columns if col not in ['date_id',
TARGET]]

X = train[features].fillna(0)
y = train[TARGET].fillna(0)

# Save feature list for later testing with
open('/kaggle/working/feature_list.pkl', 'wb') as f:
pickle.dump(features, f)

# ---------------------------
# 3) Train/Test Split
# ---------------------------
X_train, X_val, y_train, y_val = train_test_split(X, y, test_size=0.2,
random_state=42)

# ---------------------------
# 4) Scaling
# ---------------------------scaler
= StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_val_scaled = scaler.transform(X_val)

# ---------------------------
# 5) Build Model
# ---------------------------def
build_model(input_dim):     model
= tf.keras.Sequential([
        tf.keras.layers.Dense(256, activation='relu',
input_shape=(input_dim,)),
```

```
        tf.keras.layers.Dropout(0.3),
        tf.keras.layers.Dense(128, activation='relu'),
tf.keras.layers.Dropout(0.2),
        tf.keras.layers.Dense(64, activation='relu'),
        tf.keras.layers.Dense(1, activation='linear')  # regression
output     ])
    model.compile(
        optimizer=tf.keras.optimizers.Adam(learning_rate=1e-3),
loss='mse',         metrics=['mae']
    )
    return model

model = build_model(X_train_scaled.shape[1])
model.summary()

# ---------------------------
# 6) Training
# ---------------------------history
= model.fit(
    X_train_scaled, y_train,
    validation_data=(X_val_scaled, y_val),
epochs=30,     batch_size=64,
verbose=1
)

# ---------------------------
# 7) Save Model
# ---------------------------
model.save('/kaggle/working/hull_market_model.keras')
print(" Model saved!")

# ---------------------------
# 8) Evaluate on Test Dataset
# ---------------------------
# Reload model
model =
tf.keras.models.load_model('/kaggle/working/hull_market_model.keras')

# Align features with
open('/kaggle/working/feature_list.pkl', 'rb') as f:
features = pickle.load(f)

for f in features:     if f
not in test.columns:
test[f] = 0

X_test = test[features].fillna(0)
```

```
X_test_scaled = scaler.transform(X_test)

y_pred = model.predict(X_test_scaled).flatten()
y_test = test[TARGET].values if TARGET in test.columns else None

# ---------------------------
# 9) Metrics
# ---------------------------if y_test is
not None:     print("\n=== Regression
Metrics ===")
    print("MSE:", mean_squared_error(y_test, y_pred))
print("MAE:", mean_absolute_error(y_test, y_pred))
print("R2 Score:", r2_score(y_test, y_pred))

    y_test_class = (y_test > 0).astype(int)
y_pred_class = (y_pred > 0).astype(int)

    print("\n=== Classification Metrics ===")
    print("Accuracy:", accuracy_score(y_test_class, y_pred_class))
print("Precision:", precision_score(y_test_class, y_pred_class))
print("Recall:", recall_score(y_test_class, y_pred_class))
print("F1 Score:", f1_score(y_test_class, y_pred_class))
print("ROC AUC:", roc_auc_score(y_test_class, y_pred))

    # Confusion Matrix
    cm = confusion_matrix(y_test_class, y_pred_class)
plt.figure(figsize=(6,5))
    sns.heatmap(cm, annot=True, fmt="d", cmap="Blues")
plt.xlabel("Predicted")     plt.ylabel("Actual")
plt.title("Confusion Matrix")     plt.show()

    # ROC Curve
    fpr, tpr, thresholds = roc_curve(y_test_class, y_pred)
plt.figure(figsize=(6,5))
    plt.plot(fpr, tpr, label=f"AUC = {roc_auc_score(y_test_class,
y_pred):.4f}", linewidth=2)     plt.plot([0,1], [0,1], 'k--')
plt.xlabel("False Positive Rate")     plt.ylabel("True Positive
Rate")     plt.title("ROC Curve")     plt.legend()     plt.show()

print("End-to-end training + testing pipeline completed!")

Model: "sequential_2"
```
□□

```
□□□□□□
□ Layer (type)                      □ Output Shape            □
Param # □
□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□
□□□□□□
│ dense_8 (Dense)                  │ (None, 256)              │
24,832 │
├──────────────────────────────────┼──────────────────────────┼───────
──────┤
│ dropout_4 (Dropout)              │ (None, 256)              │
0 │
├──────────────────────────────────┼──────────────────────────┼───────
──────┤
│ dense_9 (Dense)                  │ (None, 128)              │
32,896 │
├──────────────────────────────────┼──────────────────────────┼───────
──────┤
│ dropout_5 (Dropout)              │ (None, 128)              │
0 │
├──────────────────────────────────┼──────────────────────────┼───────
──────┤
│ dense_10 (Dense)                 │ (None, 64)               │
8,256 │
├──────────────────────────────────┼──────────────────────────┼───────
──────┤
│ dense_11 (Dense)                 │ (None, 1)                │
65 │
└──────────────────────────────────┴──────────────────────────┴───────
──────┘

 Total params: 66,049 (258.00 KB)

 Trainable params: 66,049 (258.00 KB)  Non-

trainable params: 0 (0.00 B)

Epoch 1/30
113/113 □□□□□□□□□□□□□□□□□□□□ 4s 7ms/step - loss: 0.0854 - mae: 0.1986
- val_loss: 0.0020 - val_mae: 0.0347
Epoch 2/30
113/113 □□□□□□□□□□□□□□□□□□□□ 1s 5ms/step - loss: 0.0057 - mae: 0.0577
- val_loss: 8.4267e-04 - val_mae: 0.0211
Epoch 3/30
113/113 □□□□□□□□□□□□□□□□□□□□ 1s 5ms/step - loss: 0.0026 - mae: 0.0380
- val_loss: 4.0151e-04 - val_mae: 0.0150
Epoch 4/30
113/113 □□□□□□□□□□□□□□□□□□□□ 1s 5ms/step - loss: 0.0016 - mae: 0.0295
- val_loss: 2.4362e-04 - val_mae: 0.0117
Epoch 5/30
```

```
113/113 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - loss: 9.3903e-04 - mae:
0.0226 - val_loss: 1.8919e-04 - val_mae: 0.0105
Epoch 6/30

113/113 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - loss: 6.8076e-04 - mae:
0.0193 - val_loss: 1.7230e-04 - val_mae: 0.0098
Epoch 7/30

113/113 ━━━━━━━━━━━━━━━━━━━━ 1s 6ms/step - loss: 4.3242e-04 - mae:
0.0152 - val_loss: 1.0505e-04 - val_mae: 0.0077
Epoch 8/30

113/113 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - loss: 3.3661e-04 - mae:
0.0129 - val_loss: 8.9662e-05 - val_mae: 0.0070
Epoch 9/30

113/113 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - loss: 2.5372e-04 - mae:
0.0115 - val_loss: 8.5646e-05 - val_mae: 0.0068
Epoch 10/30

113/113 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - loss: 2.0641e-04 - mae:
0.0102 - val_loss: 8.3342e-05 - val_mae: 0.0067
Epoch 11/30

113/113 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - loss: 1.7721e-04 - mae:
0.0094 - val_loss: 8.5575e-05 - val_mae: 0.0068
Epoch 12/30

113/113 ━━━━━━━━━━━━━━━━━━━━ 1s 6ms/step - loss: 1.7306e-04 - mae:
0.0092 - val_loss: 9.3406e-05 - val_mae: 0.0073
Epoch 13/30

113/113 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - loss: 1.4139e-04 - mae:
0.0086 - val_loss: 8.0085e-05 - val_mae: 0.0065
Epoch 14/30

113/113 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - loss: 1.3062e-04 - mae:
0.0081 - val_loss: 7.9345e-05 - val_mae: 0.0066
Epoch 15/30

113/113 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - loss: 1.1160e-04 - mae:
0.0077 - val_loss: 6.9694e-05 - val_mae: 0.0061
Epoch 16/30

113/113 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - loss: 1.1264e-04 - mae:
0.0077 - val_loss: 7.2018e-05 - val_mae: 0.0063
Epoch 17/30

113/113 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - loss: 1.0531e-04 - mae:
0.0074 - val_loss: 6.7752e-05 - val_mae: 0.0061
```

```
Epoch 18/30
113/113 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - loss: 1.0052e-04 - mae:
0.0074 - val_loss: 5.8503e-05 - val_mae: 0.0057
Epoch 19/30
113/113 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - loss: 1.0722e-04 - mae:
0.0074 - val_loss: 5.4363e-05 - val_mae: 0.0055
Epoch 20/30
113/113 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - loss: 8.8928e-05 - mae:
0.0068 - val_loss: 5.2278e-05 - val_mae: 0.0054
Epoch 21/30
113/113 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - loss: 8.2362e-05 - mae:
```

```
0.0066 - val_loss: 5.1753e-05 - val_mae: 0.0055 Epoch 22/30
113/113 □□□□□□□□□□□□□□□□□□□□□□□□ 1s 5ms/step - loss: 7.4342e-05 - mae:
0.0064 - val_loss: 4.9170e-05 - val_mae: 0.0053 Epoch 23/30
113/113 □□□□□□□□□□□□□□□□□□□□□□□□ 1s 5ms/step - loss: 7.8460e-05 - mae:
0.0064 - val_loss: 4.4998e-05 - val_mae: 0.0050 Epoch 24/30
113/113 □□□□□□□□□□□□□□□□□□□□□□□□ 1s 5ms/step - loss: 7.0521e-05 - mae:
0.0062 - val_loss: 4.3808e-05 - val_mae: 0.0052 Epoch 25/30
113/113 □□□□□□□□□□□□□□□□□□□□□□□□ 1s 5ms/step - loss: 6.5178e-05 - mae:
0.0059 - val_loss: 3.6829e-05 - val_mae: 0.0045 Epoch 26/30
113/113 □□□□□□□□□□□□□□□□□□□□□□□□ 1s 5ms/step - loss: 6.1207e-05 - mae:
0.0059 - val_loss: 3.3200e-05 - val_mae: 0.0042 Epoch 27/30
113/113 □□□□□□□□□□□□□□□□□□□□□□□□ 1s 5ms/step - loss: 5.6742e-05 - mae:
0.0055 - val_loss: 3.1960e-05 - val_mae: 0.0044 Epoch 28/30
113/113 □□□□□□□□□□□□□□□□□□□□□□□□ 1s 5ms/step - loss: 5.7340e-05 - mae:
0.0056 - val_loss: 2.8100e-05 - val_mae: 0.0040 Epoch 29/30
113/113 □□□□□□□□□□□□□□□□□□□□□□□□ 1s 5ms/step - loss: 6.6945e-05 - mae:
0.0055 - val_loss: 2.8727e-05 - val_mae: 0.0039 Epoch 30/30
113/113 □□□□□□□□□□□□□□□□□□□□□□□□ 1s 5ms/step - loss: 4.9272e-05 - mae:
0.0052 - val_loss: 3.2400e-05 - val_mae: 0.0045
Model saved!
1/1 □□□□□□□□□□□□□□□□□□□□□□□□ 0s 93ms/step End-to-end
training + testing pipeline completed!
```