p5udldaf0

March 2, 2025

[1]:
```python
# 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/jane-street-real-time-market-data-forecasting/responders.csv
/kaggle/input/jane-street-real-time-market-data-
forecasting/sample_submission.csv
/kaggle/input/jane-street-real-time-market-data-forecasting/features.csv
/kaggle/input/jane-street-real-time-market-data-
forecasting/train.parquet/partition_id=4/part-0.parquet
/kaggle/input/jane-street-real-time-market-data-
forecasting/train.parquet/partition_id=5/part-0.parquet
/kaggle/input/jane-street-real-time-market-data-
forecasting/train.parquet/partition_id=6/part-0.parquet
/kaggle/input/jane-street-real-time-market-data-
forecasting/train.parquet/partition_id=3/part-0.parquet
/kaggle/input/jane-street-real-time-market-data-
forecasting/train.parquet/partition_id=1/part-0.parquet
/kaggle/input/jane-street-real-time-market-data-
```

```
forecasting/train.parquet/partition_id=8/part-0.parquet
/kaggle/input/jane-street-real-time-market-data-
forecasting/train.parquet/partition_id=2/part-0.parquet
/kaggle/input/jane-street-real-time-market-data-
forecasting/train.parquet/partition_id=0/part-0.parquet
/kaggle/input/jane-street-real-time-market-data-
forecasting/train.parquet/partition_id=7/part-0.parquet
/kaggle/input/jane-street-real-time-market-data-
forecasting/train.parquet/partition_id=9/part-0.parquet
/kaggle/input/jane-street-real-time-market-data-
forecasting/lags.parquet/date_id=0/part-0.parquet
/kaggle/input/jane-street-real-time-market-data-
forecasting/test.parquet/date_id=0/part-0.parquet
/kaggle/input/jane-street-real-time-market-data-
forecasting/kaggle_evaluation/jane_street_gateway.py
/kaggle/input/jane-street-real-time-market-data-
forecasting/kaggle_evaluation/jane_street_inference_server.py
/kaggle/input/jane-street-real-time-market-data-
forecasting/kaggle_evaluation/__init__.py
/kaggle/input/jane-street-real-time-market-data-
forecasting/kaggle_evaluation/core/templates.py
/kaggle/input/jane-street-real-time-market-data-
forecasting/kaggle_evaluation/core/base_gateway.py
/kaggle/input/jane-street-real-time-market-data-
forecasting/kaggle_evaluation/core/relay.py
/kaggle/input/jane-street-real-time-market-data-
forecasting/kaggle_evaluation/core/kaggle_evaluation.proto
/kaggle/input/jane-street-real-time-market-data-
forecasting/kaggle_evaluation/core/__init__.py
/kaggle/input/jane-street-real-time-market-data-
forecasting/kaggle_evaluation/core/generated/kaggle_evaluation_pb2.py
/kaggle/input/jane-street-real-time-market-data-
forecasting/kaggle_evaluation/core/generated/kaggle_evaluation_pb2_grpc.py
/kaggle/input/jane-street-real-time-market-data-
forecasting/kaggle_evaluation/core/generated/__init__.py
```

[2]:
```
# Install required libraries (if not already installed)
!pip install pandas numpy matplotlib seaborn scikit-learn xgboost
```

```
Requirement already satisfied: pandas in /opt/conda/lib/python3.10/site-packages
(2.2.3)
Requirement already satisfied: numpy in /opt/conda/lib/python3.10/site-packages
(1.26.4)
Requirement already satisfied: matplotlib in /opt/conda/lib/python3.10/site-
packages (3.7.5)
Requirement already satisfied: seaborn in /opt/conda/lib/python3.10/site-
packages (0.12.2)
Requirement already satisfied: scikit-learn in /opt/conda/lib/python3.10/site-
```

```
packages (1.2.2)
Requirement already satisfied: xgboost in /opt/conda/lib/python3.10/site-
packages (2.0.3)
Requirement already satisfied: python-dateutil>=2.8.2 in
/opt/conda/lib/python3.10/site-packages (from pandas) (2.9.0.post0)
Requirement already satisfied: pytz>=2020.1 in /opt/conda/lib/python3.10/site-
packages (from pandas) (2024.1)
Requirement already satisfied: tzdata>=2022.7 in /opt/conda/lib/python3.10/site-
packages (from pandas) (2024.1)
Requirement already satisfied: contourpy>=1.0.1 in
/opt/conda/lib/python3.10/site-packages (from matplotlib) (1.2.1)
Requirement already satisfied: cycler>=0.10 in /opt/conda/lib/python3.10/site-
packages (from matplotlib) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in
/opt/conda/lib/python3.10/site-packages (from matplotlib) (4.53.0)
Requirement already satisfied: kiwisolver>=1.0.1 in
/opt/conda/lib/python3.10/site-packages (from matplotlib) (1.4.5)
Requirement already satisfied: packaging>=20.0 in
/opt/conda/lib/python3.10/site-packages (from matplotlib) (21.3)
Requirement already satisfied: pillow>=6.2.0 in /opt/conda/lib/python3.10/site-
packages (from matplotlib) (10.3.0)
Requirement already satisfied: pyparsing>=2.3.1 in
/opt/conda/lib/python3.10/site-packages (from matplotlib) (3.1.2)
Requirement already satisfied: scipy>=1.3.2 in /opt/conda/lib/python3.10/site-
packages (from scikit-learn) (1.14.1)
Requirement already satisfied: joblib>=1.1.1 in /opt/conda/lib/python3.10/site-
packages (from scikit-learn) (1.4.2)
Requirement already satisfied: threadpoolctl>=2.0.0 in
/opt/conda/lib/python3.10/site-packages (from scikit-learn) (3.5.0)
Requirement already satisfied: six>=1.5 in /opt/conda/lib/python3.10/site-
packages (from python-dateutil>=2.8.2->pandas) (1.16.0)
```

[3]:

```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split, TimeSeriesSplit
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import r2_score
from xgboost import XGBRegressor
import glob

# Load and Combine Train Parquet Partitions
# train_files = glob.glob("/kaggle/input/
↪jane-street-real-time-market-data-forecasting/train.parquet/*/*.parquet")
# df_list = [pd.read_parquet(file) for file in train_files]
# train_df = pd.concat(df_list, ignore_index=True)
```

```
# print("Combined Train Data Shape:", train_df.shape)
```

[ ]:

```

```

[4]:

```
import pandas as pd

# Specify paths to only two of the partition files
train_file_1 = "/kaggle/input/jane-street-real-time-market-data-forecasting/
↪train.parquet/partition_id=0/part-0.parquet"
train_file_2 = "/kaggle/input/jane-street-real-time-market-data-forecasting/
↪train.parquet/partition_id=1/part-0.parquet"

# Load only these two files
train_df_1 = pd.read_parquet(train_file_1)
train_df_2 = pd.read_parquet(train_file_2)

# Concatenate the two DataFrames
train_df = pd.concat([train_df_1, train_df_2], ignore_index=True)

print("Optimized Combined Train Data Shape:", train_df.shape)
```

```
Optimized Combined Train Data Shape: (4748457, 92)
```

[5]:

```
train_df.columns
```

[5]:

```
Index(['date_id', 'time_id', 'symbol_id', 'weight', 'feature_00', 'feature_01',
       'feature_02', 'feature_03', 'feature_04', 'feature_05', 'feature_06',
       'feature_07', 'feature_08', 'feature_09', 'feature_10', 'feature_11',
       'feature_12', 'feature_13', 'feature_14', 'feature_15', 'feature_16',
       'feature_17', 'feature_18', 'feature_19', 'feature_20', 'feature_21',
       'feature_22', 'feature_23', 'feature_24', 'feature_25', 'feature_26',
       'feature_27', 'feature_28', 'feature_29', 'feature_30', 'feature_31',
       'feature_32', 'feature_33', 'feature_34', 'feature_35', 'feature_36',
       'feature_37', 'feature_38', 'feature_39', 'feature_40', 'feature_41',
       'feature_42', 'feature_43', 'feature_44', 'feature_45', 'feature_46',
       'feature_47', 'feature_48', 'feature_49', 'feature_50', 'feature_51',
       'feature_52', 'feature_53', 'feature_54', 'feature_55', 'feature_56',
       'feature_57', 'feature_58', 'feature_59', 'feature_60', 'feature_61',
       'feature_62', 'feature_63', 'feature_64', 'feature_65', 'feature_66',
       'feature_67', 'feature_68', 'feature_69', 'feature_70', 'feature_71',
       'feature_72', 'feature_73', 'feature_74', 'feature_75', 'feature_76',
       'feature_77', 'feature_78', 'responder_0', 'responder_1', 'responder_2',
       'responder_3', 'responder_4', 'responder_5', 'responder_6',
       'responder_7', 'responder_8'],
      dtype='object')
```

[6]:
```python
# 1. Initial Data Inspection
print("Data Shape:", train_df.shape)
print("Data Columns:", train_df.columns)
print(train_df.info())
print(train_df.head())
```

```
Data Shape: (4748457, 92)
Data Columns: Index(['date_id', 'time_id', 'symbol_id', 'weight', 'feature_00',
'feature_01',
       'feature_02', 'feature_03', 'feature_04', 'feature_05', 'feature_06',
       'feature_07', 'feature_08', 'feature_09', 'feature_10', 'feature_11',
       'feature_12', 'feature_13', 'feature_14', 'feature_15', 'feature_16',
       'feature_17', 'feature_18', 'feature_19', 'feature_20', 'feature_21',
       'feature_22', 'feature_23', 'feature_24', 'feature_25', 'feature_26',
       'feature_27', 'feature_28', 'feature_29', 'feature_30', 'feature_31',
       'feature_32', 'feature_33', 'feature_34', 'feature_35', 'feature_36',
       'feature_37', 'feature_38', 'feature_39', 'feature_40', 'feature_41',
       'feature_42', 'feature_43', 'feature_44', 'feature_45', 'feature_46',
       'feature_47', 'feature_48', 'feature_49', 'feature_50', 'feature_51',
       'feature_52', 'feature_53', 'feature_54', 'feature_55', 'feature_56',
       'feature_57', 'feature_58', 'feature_59', 'feature_60', 'feature_61',
       'feature_62', 'feature_63', 'feature_64', 'feature_65', 'feature_66',
       'feature_67', 'feature_68', 'feature_69', 'feature_70', 'feature_71',
       'feature_72', 'feature_73', 'feature_74', 'feature_75', 'feature_76',
       'feature_77', 'feature_78', 'responder_0', 'responder_1', 'responder_2',
       'responder_3', 'responder_4', 'responder_5', 'responder_6',
       'responder_7', 'responder_8'],
      dtype='object')
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4748457 entries, 0 to 4748456
Data columns (total 92 columns):
 #   Column       Dtype
---  ------       -----
 0   date_id      int16
 1   time_id      int16
 2   symbol_id    int8
 3   weight       float32
 4   feature_00   float32
 5   feature_01   float32
 6   feature_02   float32
 7   feature_03   float32
 8   feature_04   float32
 9   feature_05   float32
 10  feature_06   float32
 11  feature_07   float32
 12  feature_08   float32
 13  feature_09   int8
```

```
14  feature_10   int8
15  feature_11   int16
16  feature_12   float32
17  feature_13   float32
18  feature_14   float32
19  feature_15   float32
20  feature_16   float32
21  feature_17   float32
22  feature_18   float32
23  feature_19   float32
24  feature_20   float32
25  feature_21   float32
26  feature_22   float32
27  feature_23   float32
28  feature_24   float32
29  feature_25   float32
30  feature_26   float32
31  feature_27   float32
32  feature_28   float32
33  feature_29   float32
34  feature_30   float32
35  feature_31   float32
36  feature_32   float32
37  feature_33   float32
38  feature_34   float32
39  feature_35   float32
40  feature_36   float32
41  feature_37   float32
42  feature_38   float32
43  feature_39   float32
44  feature_40   float32
45  feature_41   float32
46  feature_42   float32
47  feature_43   float32
48  feature_44   float32
49  feature_45   float32
50  feature_46   float32
51  feature_47   float32
52  feature_48   float32
53  feature_49   float32
54  feature_50   float32
55  feature_51   float32
56  feature_52   float32
57  feature_53   float32
58  feature_54   float32
59  feature_55   float32
60  feature_56   float32
61  feature_57   float32
```

```
 62  feature_58   float32
 63  feature_59   float32
 64  feature_60   float32
 65  feature_61   float32
 66  feature_62   float32
 67  feature_63   float32
 68  feature_64   float32
 69  feature_65   float32
 70  feature_66   float32
 71  feature_67   float32
 72  feature_68   float32
 73  feature_69   float32
 74  feature_70   float32
 75  feature_71   float32
 76  feature_72   float32
 77  feature_73   float32
 78  feature_74   float32
 79  feature_75   float32
 80  feature_76   float32
 81  feature_77   float32
 82  feature_78   float32
 83  responder_0  float32
 84  responder_1  float32
 85  responder_2  float32
 86  responder_3  float32
 87  responder_4  float32
 88  responder_5  float32
 89  responder_6  float32
 90  responder_7  float32
 91  responder_8  float32
dtypes: float32(86), int16(3), int8(3)
memory usage: 1.6 GB
None
   date_id  time_id  symbol_id    weight  feature_00  feature_01  feature_02  \
0        0        0          1  3.889038         NaN         NaN         NaN
1        0        0          7  1.370613         NaN         NaN         NaN
2        0        0          9  2.285698         NaN         NaN         NaN
3        0        0         10  0.690606         NaN         NaN         NaN
4        0        0         14  0.440570         NaN         NaN         NaN

   feature_03  feature_04  feature_05  …  feature_78  responder_0  \
0         NaN         NaN    0.851033  …   -0.281498     0.738489
1         NaN         NaN    0.676961  …   -0.302441     2.965889
2         NaN         NaN    1.056285  …   -0.096792    -0.864488
3         NaN         NaN    1.139366  …   -0.296244     0.408499
4         NaN         NaN    0.955200  …    3.418133    -0.373387

   responder_1  responder_2  responder_3  responder_4  responder_5  \
```

```
0    -0.069556     1.380875     2.005353     0.186018     1.218368
1     1.190077    -0.523998     3.849921     2.626981     5.000000
2    -0.280303    -0.326697     0.375781     1.271291     0.099793
3     0.223992     2.294888     1.097444     1.225872     1.225376
4    -0.502764    -0.348021    -3.928148    -1.591366    -5.000000

   responder_6  responder_7  responder_8
0     0.775981     0.346999     0.095504
1     0.703665     0.216683     0.778639
2     2.109352     0.670881     0.772828
3     1.114137     0.775199    -1.379516
4    -3.572820    -1.089123    -5.000000

[5 rows x 92 columns]
```

[7]:
```python
# 2. Check for Missing Values
missing_values = train_df.isnull().sum()
print("Missing Values in Each Column:\n", missing_values[missing_values > 0])

# 3. Basic Statistical Summaries
print("Statistical Summary:\n", train_df.describe())
```

```
Missing Values in Each Column:
 feature_00    3182052
feature_01    3182052
feature_02    3182052
feature_03    3182052
feature_04    3182052
feature_08      16980
feature_15     134281
feature_16        104
feature_17      22499
feature_18         88
feature_19         88
feature_21    4748457
feature_26    4748457
feature_27    4748457
feature_31    4748457
feature_32      53192
feature_33      53192
feature_39     757287
feature_40      53167
feature_41     199569
feature_42     757287
feature_43      53167
feature_44     199569
feature_45     256434
feature_46     256434
```

```
feature_47         87
feature_50     715904
feature_51       5593
feature_52     156604
feature_53     715904
feature_54       5593
feature_55     156604
feature_56         88
feature_57         88
feature_58      53187
feature_62     235823
feature_63     196433
feature_64     202476
feature_65     256434
feature_66     256434
feature_73      53187
feature_74      53187
feature_75         62
feature_76         62
dtype: int64
Statistical Summary:
            date_id       time_id     symbol_id        weight    feature_00  \
count  4.748457e+06  4.748457e+06  4.748457e+06  4.748457e+06  1.566405e+06
mean   1.895017e+02  4.240000e+02  1.422099e+01  1.909972e+00 -4.319896e-01
std    9.326140e+01  2.450850e+02  1.105149e+01  1.036586e+00  9.894825e-01
min    0.000000e+00  0.000000e+00  0.000000e+00  4.405696e-01 -4.156833e+00
25%    1.150000e+02  2.120000e+02  7.000000e+00  1.213389e+00 -1.068465e+00
50%    1.970000e+02  4.240000e+02  1.200000e+01  1.689712e+00 -3.603224e-01
75%    2.690000e+02  6.360000e+02  1.700000e+01  2.288223e+00  2.607641e-01
max    3.390000e+02  8.480000e+02  3.800000e+01  8.109553e+00  2.452839e+00

         feature_01    feature_02    feature_03    feature_04    feature_05  \
count  1.566405e+06  1.566405e+06  1.566405e+06  1.566405e+06  4.748457e+06
mean   1.717309e-02 -4.323236e-01 -4.317801e-01  3.542342e-05  5.504985e-03
std    8.249986e-01  9.902229e-01  9.890876e-01  8.345563e-01  1.031042e+00
min   -3.676370e+00 -3.636927e+00 -3.943468e+00 -3.984868e+00 -2.042582e+01
25%   -4.878325e-01 -1.065548e+00 -1.064637e+00 -5.463905e-01 -4.562760e-01
50%   -2.832483e-02 -3.594359e-01 -3.592629e-01 -1.002825e-02 -3.241438e-02
75%    4.868113e-01  2.603573e-01  2.596371e-01  5.296285e-01  4.094886e-01
max    4.057066e+00  2.428777e+00  2.548314e+00  4.013427e+00  3.218666e+01

       …    feature_78   responder_0   responder_1   responder_2  \
count  …  4.748457e+06  4.748457e+06  4.748457e+06  4.748457e+06
mean   … -2.955676e-02  7.198693e-03  1.287943e-02  2.011834e-03
std    …  7.840044e-01  8.912914e-01  1.008716e+00  8.519128e-01
min    … -3.909654e+00 -5.000000e+00 -5.000000e+00 -5.000000e+00
25%    … -3.089767e-01 -2.623524e-01 -2.535336e-01 -1.843708e-01
50%    … -2.097459e-01 -3.566990e-03 -2.192864e-02 -1.158360e-03
```

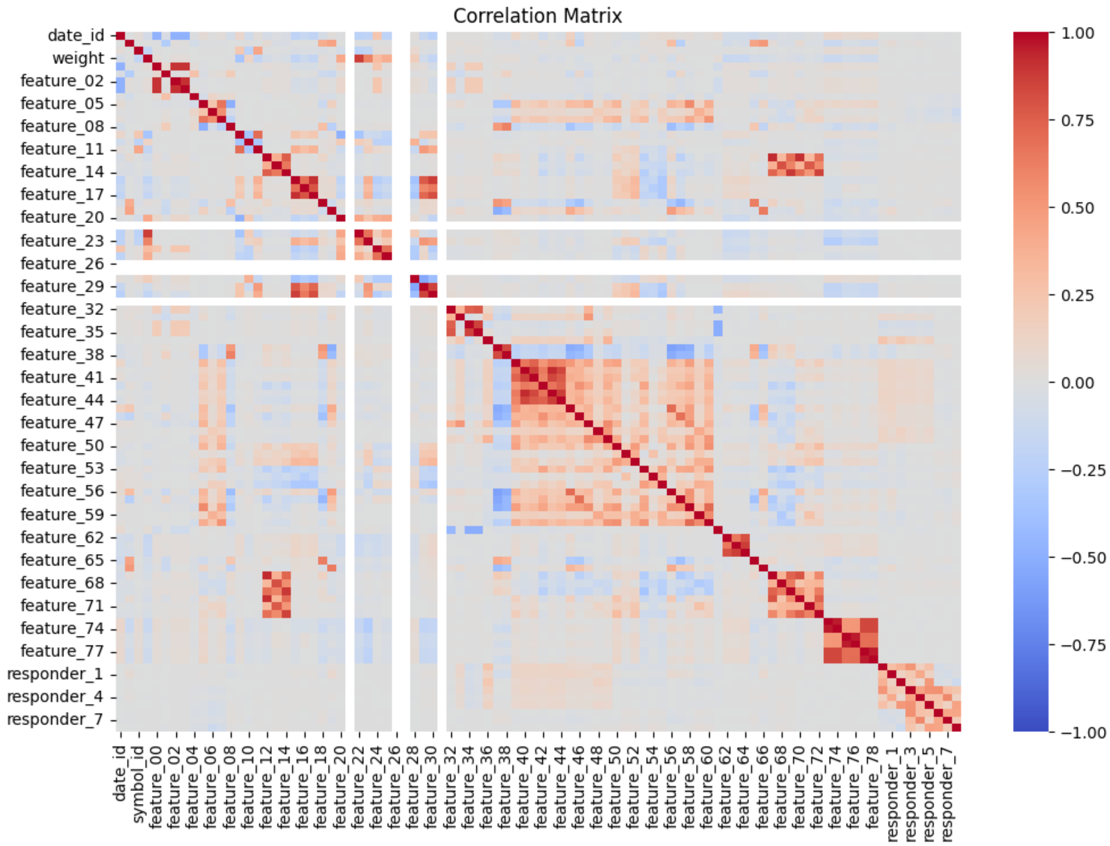
```
75%    …  3.310533e-02  2.615303e-01  2.364663e-01  1.849016e-01
max    …  7.649387e+01  5.000000e+00  5.000000e+00  5.000000e+00

        responder_3   responder_4   responder_5   responder_6   responder_7  \
count  4.748457e+06  4.748457e+06  4.748457e+06  4.748457e+06  4.748457e+06
mean   3.937299e-03  9.276975e-03 -5.818300e-04 -3.506076e-03 -8.619869e-03
std    1.081529e+00  1.150528e+00  1.008953e+00  8.884659e-01  9.274051e-01
min   -5.000000e+00 -5.000000e+00 -5.000000e+00 -5.000000e+00 -5.000000e+00
25%   -4.031982e-01 -4.890639e-01 -2.628769e-01 -3.883446e-01 -4.277329e-01
50%   -1.290329e-02 -2.222279e-02 -4.993244e-03 -1.320100e-02 -2.627048e-02
75%    3.826922e-01  4.672273e-01  2.496970e-01  3.605269e-01  3.748739e-01
max    5.000000e+00  5.000000e+00  5.000000e+00  5.000000e+00  5.000000e+00

        responder_8
count  4.748457e+06
mean  -1.592221e-03
std    8.963766e-01
min   -5.000000e+00
25%   -3.347735e-01
50%   -2.011098e-03
75%    3.255424e-01
max    5.000000e+00

[8 rows x 92 columns]
```

[8]:
```python
# 4. Correlation Analysis
correlation_matrix = train_df.corr()
plt.figure(figsize=(12, 8))
sns.heatmap(correlation_matrix, cmap="coolwarm", vmin=-1, vmax=1)
plt.title("Correlation Matrix")
plt.show()
```



Key Observations from the Correlation Matrix:

The correlation matrix highlights relationships between different features.

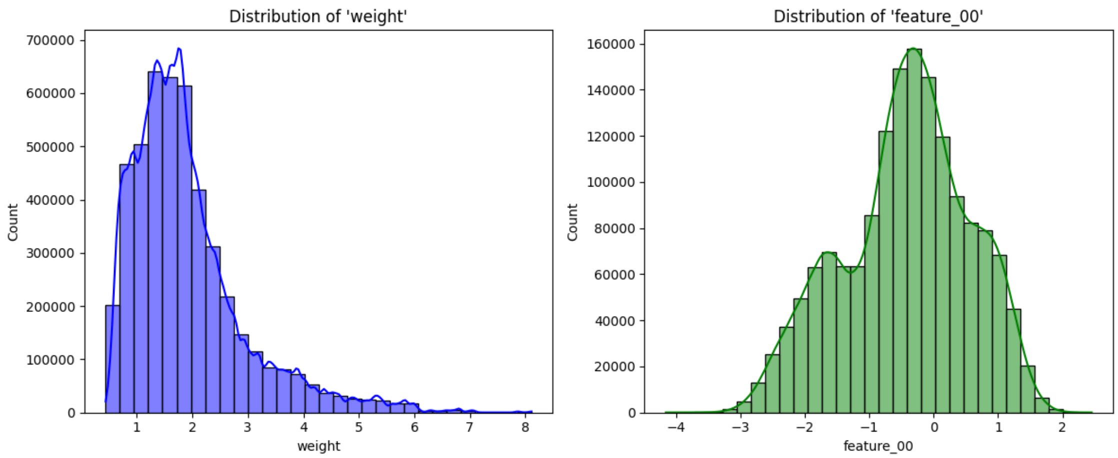
Strong correlations (deep red) suggest highly dependent variables, while strong negative correlations (deep blue) indicate inverse relationships.

The responder variables show some correlations with other features, which may be useful for predictive modeling.

Some feature blocks exhibit strong internal correlations, possibly indicating multi-collinearity.

[9]:
```
# 5. Distribution of `weight` and `feature_00`
plt.figure(figsize=(12, 5))
plt.subplot(1, 2, 1)
sns.histplot(train_df['weight'], kde=True, color='blue', bins=30)
plt.title("Distribution of 'weight'")
plt.subplot(1, 2, 2)
sns.histplot(train_df['feature_00'], kde=True, color='green', bins=30)
plt.title("Distribution of 'feature_00'")
plt.tight_layout()
plt.show()
```

```
/opt/conda/lib/python3.10/site-packages/seaborn/_oldcore.py:1119: FutureWarning:
use_inf_as_na option is deprecated and will be removed in a future version.
Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
/opt/conda/lib/python3.10/site-packages/seaborn/_oldcore.py:1119: FutureWarning:
use_inf_as_na option is deprecated and will be removed in a future version.
Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```



Weight distribution :

The weight variable is right-skewed, with most values between 1 and 3.

There are some outliers extending up to 7–8, which might require handling during preprocessing.

Feature_00_Distribution:

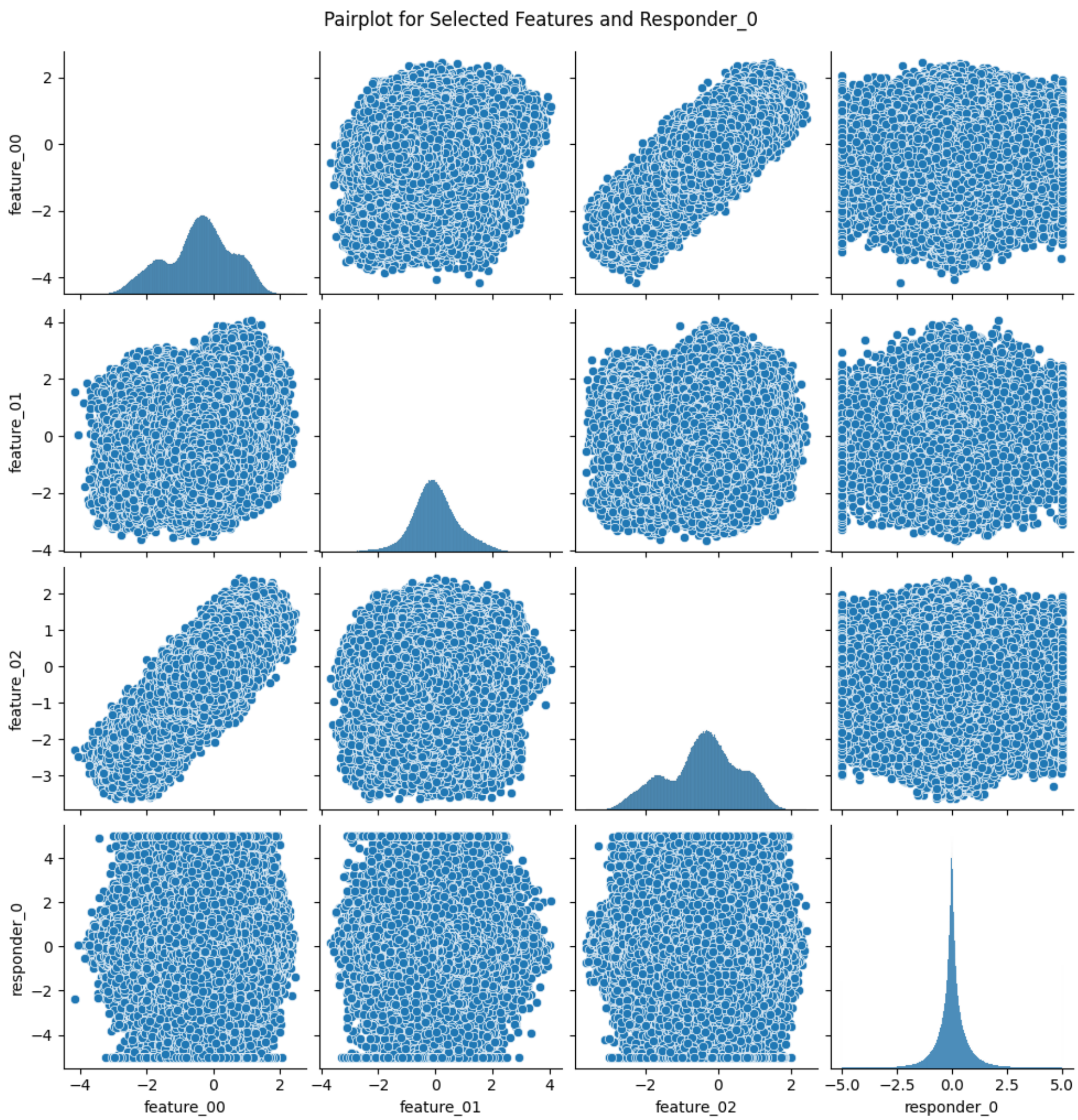
This feature follows a roughly normal distribution, centered around 0.

There are slight deviations and skewness in the tails.

[10]:
```python
# 6. Exploring Relationship between Selected Features and Responders
sns.pairplot(train_df[['feature_00', 'feature_01', 'feature_02',␣
 ↪'responder_0']])
plt.suptitle("Pairplot for Selected Features and Responder_0", y=1.02)
plt.show()
```

```
/opt/conda/lib/python3.10/site-packages/seaborn/_oldcore.py:1119: FutureWarning:
use_inf_as_na option is deprecated and will be removed in a future version.
Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
/opt/conda/lib/python3.10/site-packages/seaborn/_oldcore.py:1119: FutureWarning:
use_inf_as_na option is deprecated and will be removed in a future version.
Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```

```
/opt/conda/lib/python3.10/site-packages/seaborn/_oldcore.py:1119: FutureWarning:
use_inf_as_na option is deprecated and will be removed in a future version.
Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
/opt/conda/lib/python3.10/site-packages/seaborn/_oldcore.py:1119: FutureWarning:
use_inf_as_na option is deprecated and will be removed in a future version.
Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```



Observations:

feature_00, feature_01, and feature_02 are approximately normally distributed.

feature_00 and feature_02 show a positive correlation.

responder_0 appears uniformly distributed without strong correlation to the selected features.

[11]:
```
import pandas as pd

# Load the responder and feature CSVs
features_df = pd.read_csv("/kaggle/input/
↪jane-street-real-time-market-data-forecasting/features.csv")
responders_df = pd.read_csv("/kaggle/input/
↪jane-street-real-time-market-data-forecasting/responders.csv")

# Verify column names before merging
print("Columns in train_df:", train_df.columns)
print("Columns in responders_df:", responders_df.columns)
print("Columns in features_df:", features_df.columns)

# Merge Responders and Features with Train Data
# Adjust column names if necessary based on the print statements
try:
    train_df = train_df.merge(responders_df, on='date_id', how='left')
    train_df = train_df.merge(features_df, on='feature_id', how='left')
    print("Merged Data Shape:", train_df.shape)
except KeyError as e:
    print(f"Column not found during merge: {e}")
```

```
Columns in train_df: Index(['date_id', 'time_id', 'symbol_id', 'weight',
'feature_00', 'feature_01',
       'feature_02', 'feature_03', 'feature_04', 'feature_05', 'feature_06',
       'feature_07', 'feature_08', 'feature_09', 'feature_10', 'feature_11',
       'feature_12', 'feature_13', 'feature_14', 'feature_15', 'feature_16',
       'feature_17', 'feature_18', 'feature_19', 'feature_20', 'feature_21',
       'feature_22', 'feature_23', 'feature_24', 'feature_25', 'feature_26',
       'feature_27', 'feature_28', 'feature_29', 'feature_30', 'feature_31',
       'feature_32', 'feature_33', 'feature_34', 'feature_35', 'feature_36',
       'feature_37', 'feature_38', 'feature_39', 'feature_40', 'feature_41',
       'feature_42', 'feature_43', 'feature_44', 'feature_45', 'feature_46',
       'feature_47', 'feature_48', 'feature_49', 'feature_50', 'feature_51',
       'feature_52', 'feature_53', 'feature_54', 'feature_55', 'feature_56',
       'feature_57', 'feature_58', 'feature_59', 'feature_60', 'feature_61',
       'feature_62', 'feature_63', 'feature_64', 'feature_65', 'feature_66',
       'feature_67', 'feature_68', 'feature_69', 'feature_70', 'feature_71',
       'feature_72', 'feature_73', 'feature_74', 'feature_75', 'feature_76',
       'feature_77', 'feature_78', 'responder_0', 'responder_1', 'responder_2',
       'responder_3', 'responder_4', 'responder_5', 'responder_6',
       'responder_7', 'responder_8'],
      dtype='object')
Columns in responders_df: Index(['responder', 'tag_0', 'tag_1', 'tag_2',
```

```
'tag_3', 'tag_4'], dtype='object')
Columns in features_df: Index(['feature', 'tag_0', 'tag_1', 'tag_2', 'tag_3',
'tag_4', 'tag_5',
       'tag_6', 'tag_7', 'tag_8', 'tag_9', 'tag_10', 'tag_11', 'tag_12',
       'tag_13', 'tag_14', 'tag_15', 'tag_16'],
      dtype='object')
Column not found during merge: 'date_id'
```

[12]:

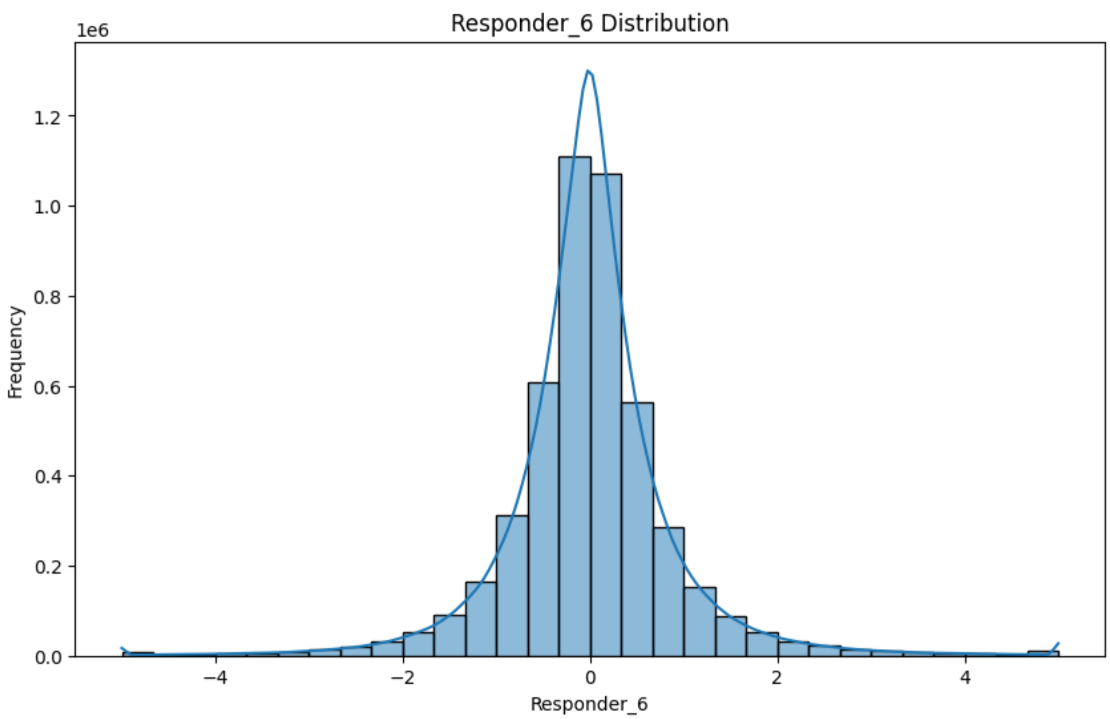
```python
# Initial Exploration
print("Dataset Columns:", train_df.columns)
print("Dataset Types:\n", train_df.dtypes)
print("Missing Values:\n", train_df.isnull().sum())

# Target Variable Distribution (Responder_6)
plt.figure(figsize=(10, 6))
sns.histplot(train_df['responder_6'], kde=True, bins=30)
plt.title('Responder_6 Distribution')
plt.xlabel('Responder_6')
plt.ylabel('Frequency')
plt.show()
```

```
Dataset Columns: Index(['date_id', 'time_id', 'symbol_id', 'weight',
'feature_00', 'feature_01',
       'feature_02', 'feature_03', 'feature_04', 'feature_05', 'feature_06',
       'feature_07', 'feature_08', 'feature_09', 'feature_10', 'feature_11',
       'feature_12', 'feature_13', 'feature_14', 'feature_15', 'feature_16',
       'feature_17', 'feature_18', 'feature_19', 'feature_20', 'feature_21',
       'feature_22', 'feature_23', 'feature_24', 'feature_25', 'feature_26',
       'feature_27', 'feature_28', 'feature_29', 'feature_30', 'feature_31',
       'feature_32', 'feature_33', 'feature_34', 'feature_35', 'feature_36',
       'feature_37', 'feature_38', 'feature_39', 'feature_40', 'feature_41',
       'feature_42', 'feature_43', 'feature_44', 'feature_45', 'feature_46',
       'feature_47', 'feature_48', 'feature_49', 'feature_50', 'feature_51',
       'feature_52', 'feature_53', 'feature_54', 'feature_55', 'feature_56',
       'feature_57', 'feature_58', 'feature_59', 'feature_60', 'feature_61',
       'feature_62', 'feature_63', 'feature_64', 'feature_65', 'feature_66',
       'feature_67', 'feature_68', 'feature_69', 'feature_70', 'feature_71',
       'feature_72', 'feature_73', 'feature_74', 'feature_75', 'feature_76',
       'feature_77', 'feature_78', 'responder_0', 'responder_1', 'responder_2',
       'responder_3', 'responder_4', 'responder_5', 'responder_6',
       'responder_7', 'responder_8'],
      dtype='object')
Dataset Types:
 date_id          int16
time_id          int16
symbol_id         int8
weight         float32
feature_00     float32
```

```
                …
responder_4    float32
responder_5    float32
responder_6    float32
responder_7    float32
responder_8    float32
Length: 92, dtype: object
Missing Values:
 date_id              0
time_id              0
symbol_id            0
weight               0
feature_00     3182052
                …
responder_4          0
responder_5          0
responder_6          0
responder_7          0
responder_8          0
Length: 92, dtype: int64
```

```
/opt/conda/lib/python3.10/site-packages/seaborn/_oldcore.py:1119: FutureWarning:
use_inf_as_na option is deprecated and will be removed in a future version.
Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```



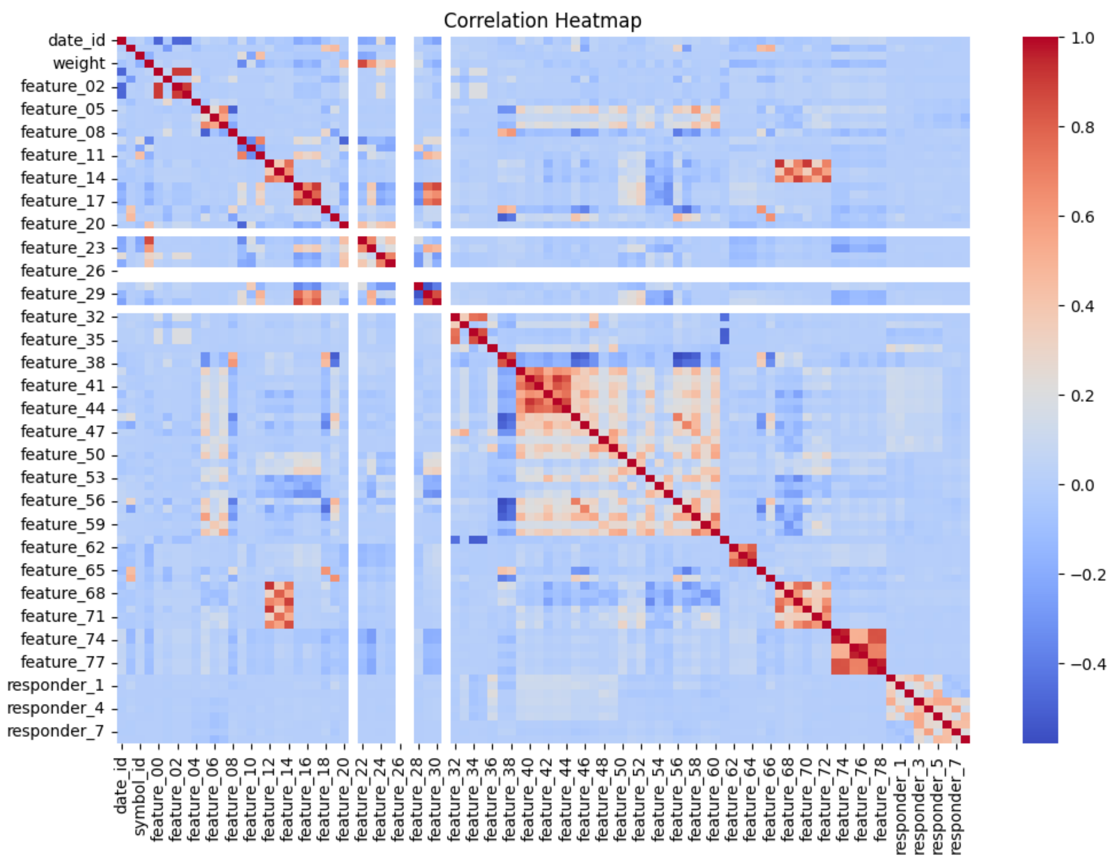
Key Observations:

The distribution follows a normal-like pattern, centered around zero, with most values concentrated near the mean.

The symmetric shape suggests that Responder_6 does not have strong skewness.

The KDE overlay reinforces the assumption of a Gaussian distribution, which is useful for statistical modeling.

[13]:
```python
# Correlation Analysis
corr_matrix = train_df.corr()
plt.figure(figsize=(12, 8))
sns.heatmap(corr_matrix, cmap='coolwarm', annot=False, fmt=".2f")
plt.title("Correlation Heatmap")
plt.show()
```

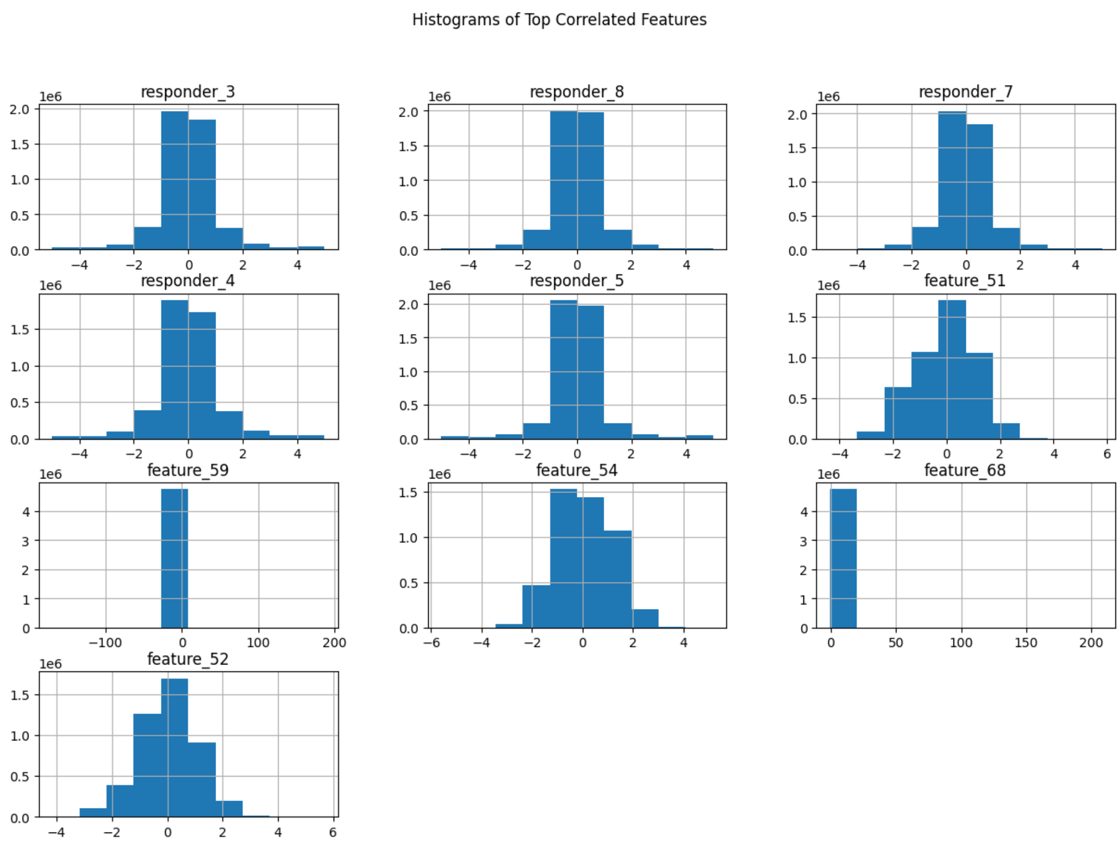


[14]:
```python
# Feature Selection by Correlation
correlation_target = corr_matrix['responder_6'].sort_values(ascending=False)
top_features = correlation_target.index[1:11]  # Top 10 correlated features, excluding responder_6 itself
```

```
print("Top Correlated Features with responder_6:\n", correlation_target.
    ↪head(10))
```

```
Top Correlated Features with responder_6:
 responder_6     1.000000
responder_3     0.542425
responder_8     0.441470
responder_7     0.434857
responder_4     0.268392
responder_5     0.248837
feature_51      0.028686
feature_59      0.021348
feature_54      0.021329
feature_68      0.020753
Name: responder_6, dtype: float64
```

[15]:
```
# Feature Distributions
train_df[top_features].hist(figsize=(15, 10))
plt.suptitle("Histograms of Top Correlated Features")
plt.show()
```



Observations:

Most features exhibit a normal distribution, centered around zero.

Some features, like feature_59 and feature_68, show distinct patterns that may require further investigation.

Understanding these distributions can help in feature engineering and model optimization.

[ ]:

```

```