FoML Hackathon 2023

Name-Sanyam Kaul, Roll No-CS23MTECH14011

Name-Mayuresh Rajesh Dindorkar, Roll No-CS23MTECH14007

In [23]:

```
import pandas as pd
import numpy as np
from sklearn.metrics import f1_score, accuracy_score
from sklearn.impute import KNNImputer
import matplotlib.pyplot as plt
from random import randint
import seaborn as sns
from sklearn.model_selection import RandomizedSearchCV
from sklearn.ensemble import GradientBoostingClassifier, RandomForestClassifier, Vo

import warnings
warnings.filterwarnings("ignore")
```

Reading the dataset

In [24]:

```
train_set = pd.read_csv('iith_foml_2023_train.csv')
test_set = pd.read_csv('iith_foml_2023_test.csv')
```

In [25]:

```
print('TrainSet: ', train_set.shape)
print('TestSet: ', test_set.shape)
```

```
TrainSet:  (994, 25)
TestSet:  (426, 24)
```

In [26]:

```
train_set.head()
```

Out[26]:

	Feature 1 (Discrete)	Feature 2 (Discrete)	Feature 3 (Discrete)	Feature 4 (Discrete)	Feature 5 (Discrete)	Feature 6 (Discrete)	Feature 7 (Discrete)	Feature 8 (Discrete)	Feature 9
0	1404	12	64	14	3	1	1	1	110.502
1	909	0	235	32	1	1	1	1	-40.448
2	654	3	175	2	1	1	1	1	-27.445
3	1372	12	382	14	2	0	1	0	0.001
4	786	3	199	2	1	0	1	0	0.001

5 rows × 25 columns

In [27]:

```
test_set.head()
```

Out[27]:

	Feature 1 (Discrete)	Feature 2 (Discrete)	Feature 3 (Discrete)	Feature 4 (Discrete)	Feature 5 (Discrete)	Feature 6 (Discrete)	Feature 7 (Discrete)	Feature 8 (Discrete)	Feature 9
0	146	12	42	14	7	1	1	1	118.004
1	35	0	12	5	0	0	1	0	0.001
2	1018	8	259	2	1	1	1	1	NaN
3	383	7	117	5	1	1	1	1	53.002
4	1216	7	40	5	2	0	1	4	0.005

5 rows × 24 columns

In [28]:

```
X_train = train_set.drop('Target Variable (Discrete)', axis=1)
Y_train = train_set['Target Variable (Discrete)']
X_test = test_set
print('X_train: ',X_train.shape)
print('Y_train: ',Y_train.shape)
```

```
X_train:  (994, 24)
Y_train:  (994,)
```

In [29]:

```
def pred_and_save_to_csv(X, clf, filename):
    # Make predictions on the test set
    y_pred = clf.predict(X)
    # Create a DataFrame with "sequence_no" and "prediction" columns
    results_df = pd.DataFrame({'id': X.index + 1, 'Category': y_pred})

    # Save the DataFrame to a CSV file
    results_df.to_csv(filename, index=False)
```

Analyzing Training Set: -

In [30]:

```
X_train.isnull().sum()
```

Out[30]:

```
Feature 1 (Discrete)       0
Feature 2 (Discrete)       0
Feature 3 (Discrete)       0
Feature 4 (Discrete)       0
Feature 5 (Discrete)       0
Feature 6 (Discrete)       0
Feature 7 (Discrete)       0
Feature 8 (Discrete)       0
Feature 9                 14
Feature 10                 1
Feature 11                 1
Feature 12                 1
Feature 13                 1
Feature 14                 1
Feature 15                72
Feature 16               669
Feature 17               546
Feature 18               330
Feature 19 (Discrete)      0
Feature 20 (Discrete)      0
Feature 21 (Discrete)      0
Feature 22 (Discrete)      0
Feature 23 (Discrete)      0
Feature 24                 1
dtype: int64
```

In [31]:

```
X_test.isnull().sum()
```

Out[31]:

```
Feature 1 (Discrete)       0
Feature 2 (Discrete)       0
Feature 3 (Discrete)       0
Feature 4 (Discrete)       0
Feature 5 (Discrete)       0
Feature 6 (Discrete)       0
Feature 7 (Discrete)       0
Feature 8 (Discrete)       0
Feature 9                  4
Feature 10                 0
Feature 11                 0
Feature 12                 0
Feature 13                 0
Feature 14                 2
Feature 15                31
Feature 16               279
Feature 17               225
Feature 18               114
Feature 19 (Discrete)      0
Feature 20 (Discrete)      0
Feature 21 (Discrete)      0
Feature 22 (Discrete)      0
Feature 23 (Discrete)      0
Feature 24                 0
dtype: int64
```

We can observe that more than half of the the data points in the feaure columns 'Feature 16' and 'Feature 17' are missing.
Hence, dropping the columns

In [32]:

```
X_train.drop(['Feature 16', 'Feature 17'], axis=1, inplace=True)
X_test.drop(['Feature 16', 'Feature 17'], axis=1, inplace=True)
```

Imputing the missing values with KnnImputer

In [33]:

```
knn_imputer = KNNImputer(n_neighbors=5)
X_train = pd.DataFrame(knn_imputer.fit_transform(X_train), columns=X_train.columns)
X_test = pd.DataFrame(knn_imputer.transform(X_test), columns=X_test.columns)
```

In [34]:

```
X_train.isnull().sum()
```

Out[34]:

```
Feature 1 (Discrete)     0
Feature 2 (Discrete)     0
Feature 3 (Discrete)     0
Feature 4 (Discrete)     0
Feature 5 (Discrete)     0
Feature 6 (Discrete)     0
Feature 7 (Discrete)     0
Feature 8 (Discrete)     0
Feature 9                0
Feature 10               0
Feature 11               0
Feature 12               0
Feature 13               0
Feature 14               0
Feature 15               0
Feature 18               0
Feature 19 (Discrete)    0
Feature 20 (Discrete)    0
Feature 21 (Discrete)    0
Feature 22 (Discrete)    0
Feature 23 (Discrete)    0
Feature 24               0
dtype: int64
```

In [35]:

```
X_test.isnull().sum()
```

Out[35]:

```
Feature 1 (Discrete)     0
Feature 2 (Discrete)     0
Feature 3 (Discrete)     0
Feature 4 (Discrete)     0
Feature 5 (Discrete)     0
Feature 6 (Discrete)     0
Feature 7 (Discrete)     0
Feature 8 (Discrete)     0
Feature 9                0
Feature 10               0
Feature 11               0
Feature 12               0
Feature 13               0
Feature 14               0
Feature 15               0
Feature 18               0
Feature 19 (Discrete)    0
Feature 20 (Discrete)    0
Feature 21 (Discrete)    0
Feature 22 (Discrete)    0
Feature 23 (Discrete)    0
Feature 24               0
dtype: int64
```

Calculating correlation among feature columns

In [36]:

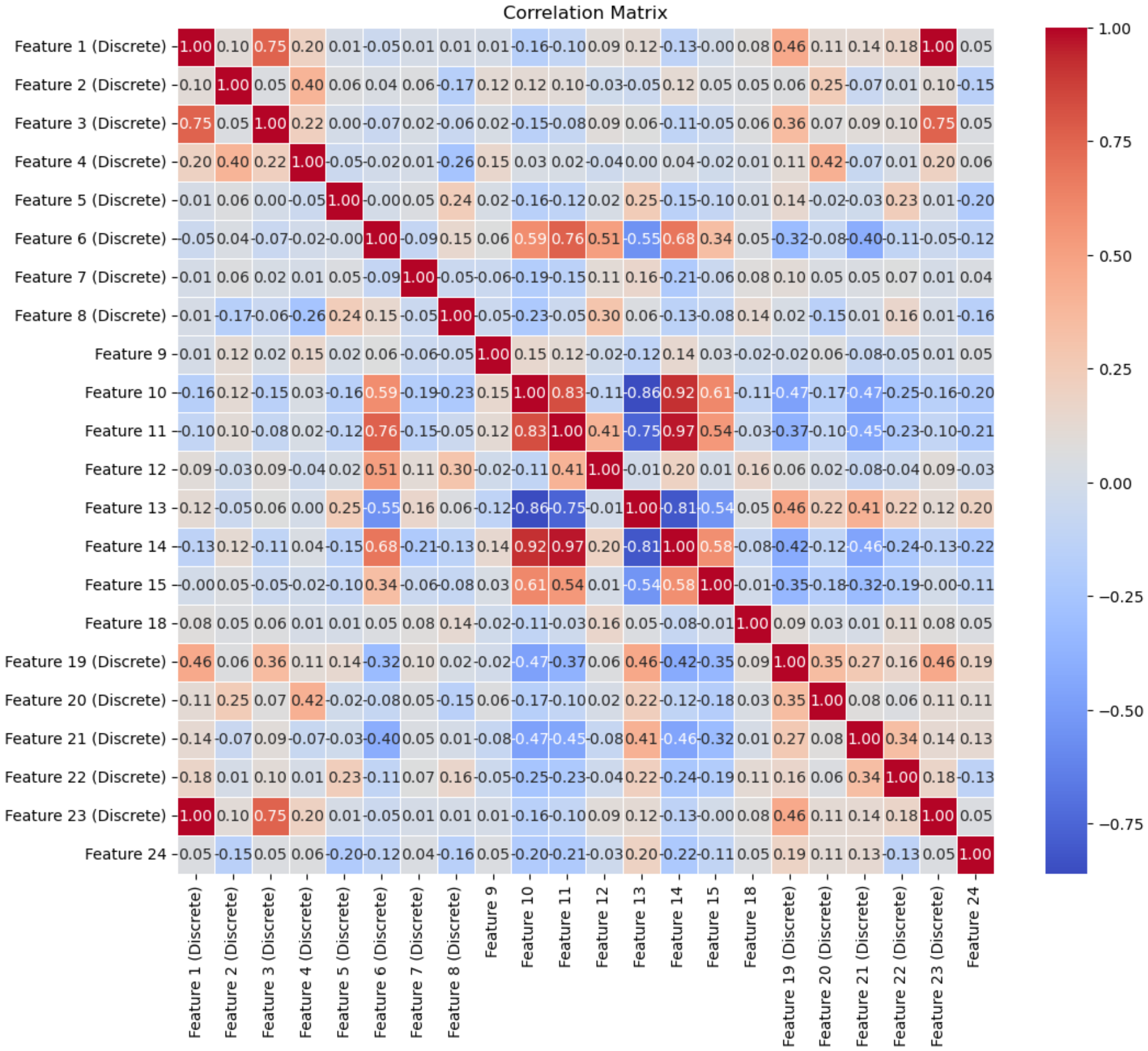
```
# With the following function we can select highly correlated features
# it will remove the first feature that is correlated with anything other feature

def correlation(dataset, threshold):
    col_corr = set()  # Set of all the names of correlated columns
    corr_matrix = dataset.corr()
    for i in range(len(corr_matrix.columns)):
        for j in range(i):
            if abs(corr_matrix.iloc[i, j]) > threshold: # we are interested in abso
                colname = corr_matrix.columns[i]  # getting the name of column
                col_corr.add(colname)
    return col_corr
```

In [37]:

```
# Generate a correlation matrix
correlation_matrix = X_train.corr()

# Create a heatmap to visualize the correlation matrix
plt.figure(figsize=(12, 10))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt=".2f", linewidths=
plt.title('Correlation Matrix')
plt.show()
```



Dropping the columns having correlation > 0.70

In [38]:

```
corr_features = correlation(X_train, 0.70)
corr_features
```

Out[38]:

```
{'Feature 11',
 'Feature 13',
 'Feature 14',
 'Feature 23 (Discrete)',
 'Feature 3 (Discrete)'}
```

In [39]:

```
X_train.drop(corr_features,axis=1, inplace=True)
X_test.drop(corr_features,axis=1, inplace = True)
```

In [40]:

```
print(X_train.shape)
print(X_test.shape)
```

```
(994, 17)
(426, 17)
```

Voting

In [41]:

```
voting_classifier = VotingClassifier(estimators=[
    ('gb', GradientBoostingClassifier(learning_rate=0.1, max_depth=6, max_features=
    ('rf', RandomForestClassifier(n_estimators = 580,min_samples_split=5, random_st
], voting='hard')

voting_classifier.fit(X_train, Y_train)

# Make predictions on the test set
y_pred_voting = voting_classifier.predict(X_train)

# Evaluate the accuracy of the stacking classifier
accuracy = accuracy_score(Y_train, y_pred_voting)
print(f'Train Accuracy: {accuracy}')
```

```
Train Accuracy: 1.0
```

Saving prediction to CSV

In [42]:

```
pred_and_save_to_csv(X_test, voting_classifier, 'submission.csv')
```

To take test_input.csv file and give prediction in test_output.csv

In [43]:

```
test_input = pd.read_csv('test_input.csv')
X_test_input = test_input

# Performing data-cleaning
X_test_input.drop(['Feature 16', 'Feature 17'], axis=1, inplace=True)
X_test_input = pd.DataFrame(knn_imputer.transform(X_test_input), columns=X_test_inp
X_test_input.drop(corr_features,axis=1, inplace = True)

# Getting prediction
y_pred =voting_classifier.predict(X_test_input)
pred_and_save_to_csv(X_test, voting_classifier, 'test_output.csv')
```