FoML Hackathon 2023

Name-Sanyam Kaul, Roll No-CS23MTECH14011

Name-Mayuresh Rajesh Dindorkar, Roll No-CS23MTECH14007

In [57]:

```
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

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

Reading the dataset

In [58]:

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

In [59]:

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

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

In [60]:

```
train_set.head()
```

Out[60]:

	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 [61]:

```
test_set.head()
```

Out[61]:

	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 [62]:

```
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 [63]:

```
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 [64]:

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

Out[64]:

```
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 [65]:

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

Out[65]:

```
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 [66]:

```
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 [67]:

```
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 [68]:

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

Out[68]:

```
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 [69]:

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

Out[69]:

```
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 [70]:

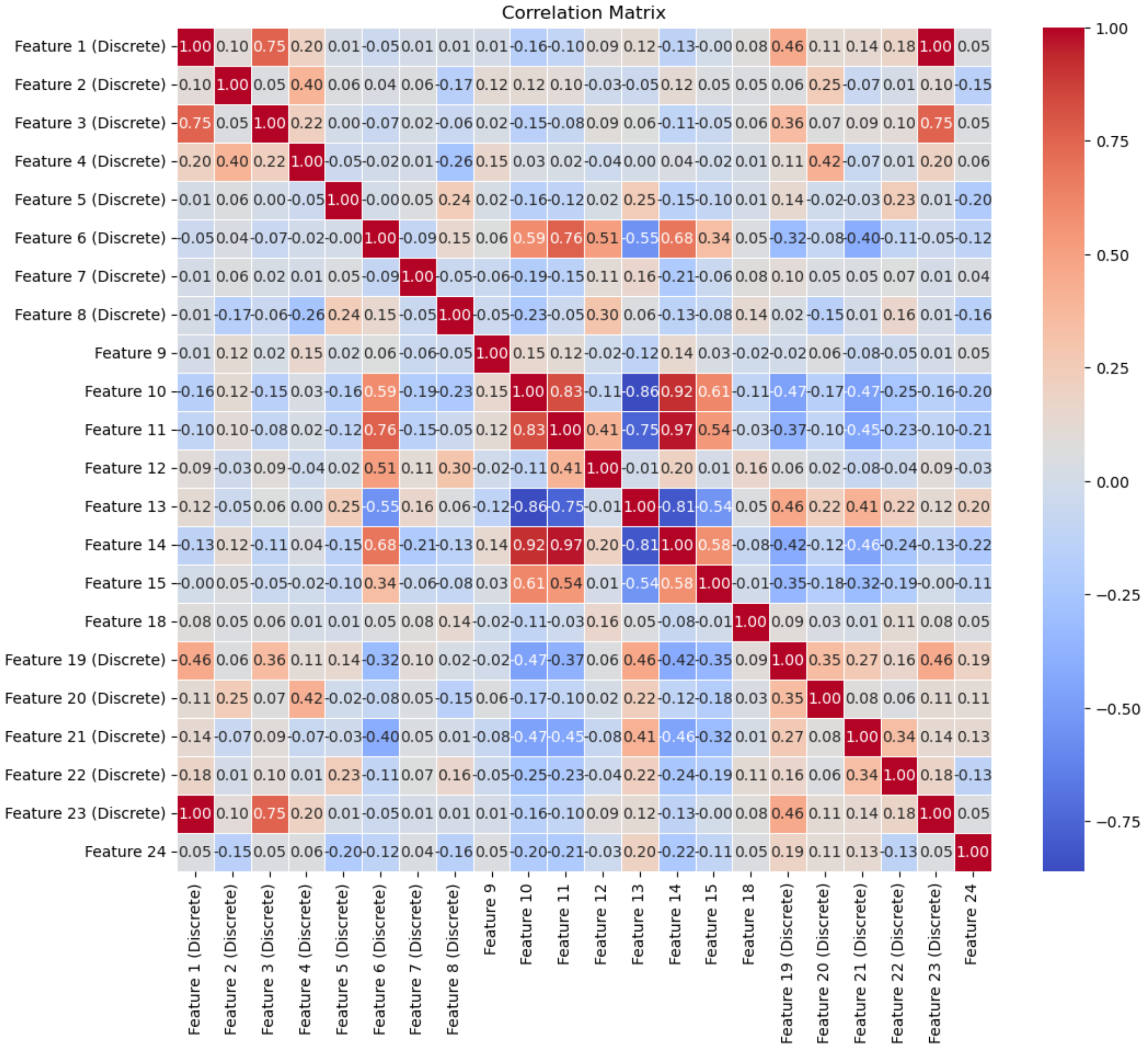
```
# With the following function we can select highly correlated features
# it will remove the first feature that is correlated with anything other feature

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 [71]:

```
# 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()
```


Correlation Matrix

	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	Feature 10	Feature 11	Feature 12	Feature 13	Feature 14	Feature 15	Feature 18	Feature 19 (Discrete)	Feature 20 (Discrete)	Feature 21 (Discrete)	Feature 22 (Discrete)	Feature 23 (Discrete)	Feature 24
Feature 1 (Discrete)	1.00	0.10	0.75	0.20	0.01	-0.05	0.01	0.01	0.01	-0.16	-0.10	0.09	0.12	-0.13	-0.00	0.08	0.46	0.11	0.14	0.18	1.00	0.05
Feature 2 (Discrete)	0.10	1.00	0.05	0.40	0.06	0.04	0.06	-0.17	0.12	0.12	0.10	-0.03	-0.05	0.12	0.05	0.05	0.06	0.25	-0.07	0.01	0.10	-0.15
Feature 3 (Discrete)	0.75	0.05	1.00	0.22	0.00	-0.07	0.02	-0.06	0.02	-0.15	-0.08	0.09	0.06	-0.11	-0.05	0.06	0.36	0.07	0.09	0.10	0.75	0.05
Feature 4 (Discrete)	0.20	0.40	0.22	1.00	-0.05	-0.02	0.01	-0.26	0.15	0.03	0.02	-0.04	0.00	0.04	-0.02	0.01	0.11	0.42	-0.07	0.01	0.20	0.06
Feature 5 (Discrete)	0.01	0.06	0.00	-0.05	1.00	-0.00	0.05	0.24	0.02	-0.16	-0.12	0.02	0.25	-0.15	-0.10	0.01	0.14	-0.02	-0.03	0.23	0.01	-0.20
Feature 6 (Discrete)	-0.05	0.04	-0.07	-0.02	-0.00	1.00	-0.09	0.15	0.06	0.59	0.76	0.51	-0.55	0.68	0.34	0.05	-0.32	-0.08	-0.40	-0.11	-0.05	-0.12
Feature 7 (Discrete)	0.01	0.06	0.02	0.01	0.05	-0.09	1.00	-0.05	-0.06	-0.19	-0.15	0.11	0.16	-0.21	-0.06	0.08	0.10	0.05	0.05	0.07	0.01	0.04
Feature 8 (Discrete)	0.01	-0.17	-0.06	-0.26	0.24	0.15	-0.05	1.00	-0.05	-0.23	-0.05	0.30	0.06	-0.13	-0.08	0.14	0.02	-0.15	0.01	0.16	0.01	-0.16
Feature 9	0.01	0.12	0.02	0.15	0.02	0.06	-0.06	-0.05	1.00	0.15	0.12	-0.02	-0.12	0.14	0.03	-0.02	-0.02	0.06	-0.08	-0.05	0.01	0.05
Feature 10	-0.16	0.12	-0.15	0.03	-0.16	0.59	-0.19	-0.23	0.15	1.00	0.83	-0.11	-0.86	0.92	0.61	-0.11	-0.47	-0.17	-0.47	-0.25	-0.16	-0.20
Feature 11	-0.10	0.10	-0.08	0.02	-0.12	0.76	-0.15	-0.05	0.12	0.83	1.00	0.41	-0.75	0.97	0.54	-0.03	-0.37	-0.10	-0.45	-0.23	-0.10	-0.21
Feature 12	0.09	-0.03	0.09	-0.04	0.02	0.51	0.11	0.30	-0.02	-0.11	0.41	1.00	-0.01	0.20	0.01	0.16	0.06	0.02	-0.08	-0.04	0.09	-0.03
Feature 13	0.12	-0.05	0.06	0.00	0.25	-0.55	0.16	0.06	-0.12	-0.86	-0.75	-0.01	1.00	-0.81	-0.54	0.05	0.46	0.22	0.41	0.22	0.12	0.20
Feature 14	-0.13	0.12	-0.11	0.04	-0.15	0.68	-0.21	-0.13	0.14	0.92	0.97	0.20	-0.81	1.00	0.58	-0.08	-0.42	-0.12	-0.46	-0.24	-0.13	-0.22
Feature 15	-0.00	0.05	-0.05	-0.02	-0.10	0.34	-0.06	-0.08	0.03	0.61	0.54	0.01	-0.54	0.58	1.00	-0.01	-0.35	-0.18	-0.32	-0.19	-0.00	-0.11
Feature 18	0.08	0.05	0.06	0.01	0.01	0.05	0.08	0.14	-0.02	-0.11	-0.03	0.16	0.05	-0.08	-0.01	1.00	0.09	0.03	0.01	0.11	0.08	0.05
Feature 19 (Discrete)	0.46	0.06	0.36	0.11	0.14	-0.32	0.10	0.02	-0.02	-0.47	-0.37	0.06	0.46	-0.42	-0.35	0.09	1.00	0.35	0.27	0.16	0.46	0.19
Feature 20 (Discrete)	0.11	0.25	0.07	0.42	-0.02	-0.08	0.05	-0.15	0.06	-0.17	-0.10	0.02	0.22	-0.12	-0.18	0.03	0.35	1.00	0.08	0.06	0.11	0.11
Feature 21 (Discrete)	0.14	-0.07	0.09	-0.07	-0.03	-0.40	0.05	0.01	-0.08	-0.47	-0.45	-0.08	0.41	-0.46	-0.32	0.01	0.27	0.08	1.00	0.34	0.14	0.13
Feature 22 (Discrete)	0.18	0.01	0.10	0.01	0.23	-0.11	0.07	0.16	-0.05	-0.25	-0.23	-0.04	0.22	-0.24	-0.19	0.11	0.16	0.06	0.34	1.00	0.18	-0.13
Feature 23 (Discrete)	1.00	0.10	0.75	0.20	0.01	-0.05	0.01	0.01	0.01	-0.16	-0.10	0.09	0.12	-0.13	-0.00	0.08	0.46	0.11	0.14	0.18	1.00	0.05
Feature 24	0.05	-0.15	0.05	0.06	-0.20	-0.12	0.04	-0.16	0.05	-0.20	-0.21	-0.03	0.20	-0.22	-0.11	0.05	0.19	0.11	0.13	-0.13	0.05	1.00

Dropping the columns having correlation > 0.70

In [72]:

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

Out[72]:

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

In [73]:
```
X_train.drop(corr_features,axis=1, inplace=True)
X_test.drop(corr_features,axis=1, inplace = True)
```

In [74]:
```
print(X_train.shape)
print(X_test.shape)
```

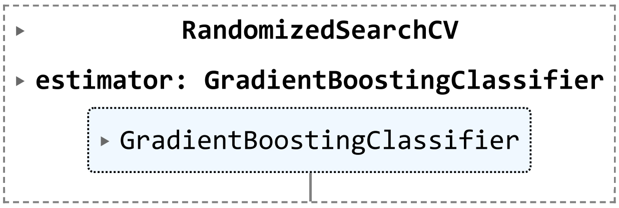
```
(994, 17)
(426, 17)
```

Gradient boost

In [75]:
```
param_dist = {
    'n_estimators': [400],
    'learning_rate': [0.1],
    'max_depth': randint(1, 10),
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4],
    'max_features': ['auto', 'sqrt', None]
}

# Create the RandomizedSearchCV object
random_search_gb = RandomizedSearchCV(estimator=GradientBoostingClassifier(), param
random_search_gb
```

Out[75]:



```
▸ RandomizedSearchCV
▸ estimator: GradientBoostingClassifier
  ▸ GradientBoostingClassifier
```

Learning hyper-parameters by RandomizedSearchCV

In [76]:
```
# # Train the classifier
# random_search_gb.fit(X_train, Y_train)

# best_params = random_search_gb.best_params_
# print("Best Hyperparameters:", best_params)
```

Fitting the Gradient Boosting model by using the learned hyper-parameters

In [77]:
```
gb = GradientBoostingClassifier(learning_rate=0.1, max_depth=6, max_features='sqrt'
gb.fit(X_train, Y_train)
y_pred_gb =gb.predict(X_train)
accuracy = accuracy_score(Y_train, y_pred_gb)
print("Train Accuracy with GrBoost Classifier:", accuracy)
```

```
Train Accuracy with GrBoost Classifier: 1.0
```

Saving prediction to CSV

In [78]:
```
pred_and_save_to_csv(X_test, gb, 'submission.csv')
```

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

In [84]:
```
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 =gb.predict(X_test_input)
pred_and_save_to_csv(X_test, gb, 'test_output.csv')
```