# *REPORT ON THE INVESTIGATION*

University of Essex

*Monzur, Rifat ID# 2200367*

*Word Count 550*

# Table of Contents

# Task 2

## Data analysis & Preprocessing

Data contains 22 columns and 1000 rows. All the columns have all values except 'F21'. F21 has 50 percent value missing. Target output class have two values: True and False. 3 columns only contains 2 type of values are F1, F11 & F14. We converted target class bool value to numeric value.

## Imputation

As F21 is missing half of the value, we need to take care of it. We applied various imputation procedures. Removing F21 column, used KNN imputer to replace missing value, used MICE imputer to replace value, used mean to replace value and also used median to replace value.

## Model Selection

Used various models like KNN, Naïve Bayes, SVC, NuSVC, Decision Tree, Random Forest, Ada Boost Classifier and Gradient Boosting. After that we go forward with model with best accuracy.

| Model | Training Score | Test Score |
|---|---|---|
| KNN | 0.810 | 0.655 |
| GaussianNB | 0.537 | 0.435 |
| BernoulliNB | 0.421 | 0.490 |
| SVC | 0.517 | 0.460 |
| NuSVC | 0.564 | 0.455 |
| Decision Tree | 1.000 | 0.830 |
| Random Forest | 1.000 | 0.875 |
| Ada Boost | 0.835 | 0745 |
| Gradient Boost | 0.959 | 0.875 |

## Grid Search for Hyper Tuning

After model selection, we did hyper tuning with grid search cross validation. We did on data without F21 column. We drop F21 column for this test.

| Model | Training Score | Test Score | Test Recall | Test Precision |
|---|---|---|---|---|
| Decision Tree | 0.924 | 0.865 | 0.870 | 0.842 |
| Logistic Regression | 0.694 | 0.715 | 0.652 | 0.706 |
| KNN | 1.000 | 0.675 | 0.663 | 0.642 |
| GaussianNB | 0.625 | 0.580 | 0.576 | 0.541 |
| Gradient Boost | 1.000 | 0.920 | 0.935 | 0.896 |

## Best Score after hyper tuning

Got the best output from grid search and some manual tuning

| Model | Training Score | Test Score | Test Recall | Test Precision |
|---|---|---|---|---|
| Decision Tree | 0.922 | 0.900 | 0.946 | 0.853 |
| Logistic Regression | 0.694 | 0.715 | 0.652 | 0.706 |
| KNN | 1.000 | 0.675 | 0,663 | 0.642 |
| GaussianNB | 0.625 | 0.580 | 0.576 | 0.541 |
| Gradient Boost | 1.000 | 0.930 | 0.957 | 0.898 |

## Result with KNN Imputer

Among all the imputers Knn imputer gave the best accuracy.

```
DecisionTreeClassifier()
{'criterion': 'entropy', 'max_depth': 5, 'min_samples_leaf': 1, 'min_samples_split': 2}
Train score: 0.922
Train Recall score 0.954, Precision score: 0.902
Test score: 0.900
Test Recall score 0.946, Precision score: 0.853
LogisticRegression()
{'max_iter': 180, 'penalty': 'l1', 'solver': 'saga'}
Train score: 0.708
Train Recall score 0.708, Precision score: 0.722
Test score: 0.740
Test Recall score 0.793, Precision score: 0.689
KNeighborsClassifier()
{'algorithm': 'auto', 'leaf_size': 1, 'metric': 'manhattan', 'n_neighbors': 37, 'weights': 'distance'}
Train score: 1.000
Train Recall score 1.000, Precision score: 1.000
Test score: 0.665
Test Recall score 0.685, Precision score: 0.624
GaussianNB()
{'var_smoothing': 0.01519911082952933}
Train score: 0.657
Train Recall score 0.705, Precision score: 0.658
Test score: 0.635
Test Recall score 0.696, Precision score: 0.587
GradientBoostingClassifier()
{'learning_rate': 0.125, 'max_depth': 5, 'n_estimators': 50}
Train score: 1.000
Train Recall score 1.000, Precision score: 1.000
Test score: 0.915
Test Recall score 0.946, Precision score: 0.879
```

## Results with MICE Imputer

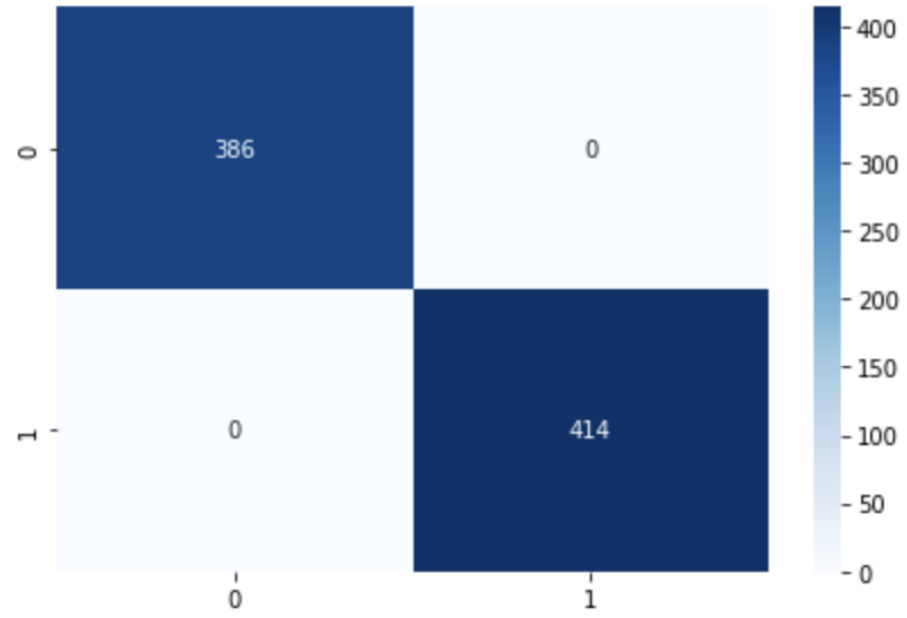
Mice imputer also gave good results. However median and mean imputer did not give good result. We also tried dropping rows with missing value. That also did not workout. It gave the worst result.

```
DecisionTreeClassifier()
{'criterion': 'entropy', 'max_depth': 5, 'min_samples_leaf': 1, 'min_samples_split': 2}
Train score: 0.919
Train Recall score 0.961, Precision score: 0.890
Test score: 0.890
Test Recall score 0.946, Precision score: 0.837
LogisticRegression()
{'max_iter': 180, 'penalty': 'l1', 'solver': 'saga'}
Train score: 0.731
Train Recall score 0.710, Precision score: 0.756
Test score: 0.750
Test Recall score 0.783, Precision score: 0.706
KNeighborsClassifier()
{'algorithm': 'auto', 'leaf_size': 1, 'metric': 'manhattan', 'n_neighbors': 37, 'weights': 'distance'}
Train score: 1.000
Train Recall score 1.000, Precision score: 1.000
Test score: 0.675
Test Recall score 0.696, Precision score: 0.634
GaussianNB()
{'var_smoothing': 0.01519911082952933}
Train score: 0.679
Train Recall score 0.691, Precision score: 0.689
Test score: 0.635
Test Recall score 0.663, Precision score: 0.592
GradientBoostingClassifier()
{'learning_rate': 0.125, 'max_depth': 5, 'n_estimators': 50}
Train score: 0.999
Train Recall score 0.998, Precision score: 1.000
Test score: 0.910
Test Recall score 0.935, Precision score: 0.878
```

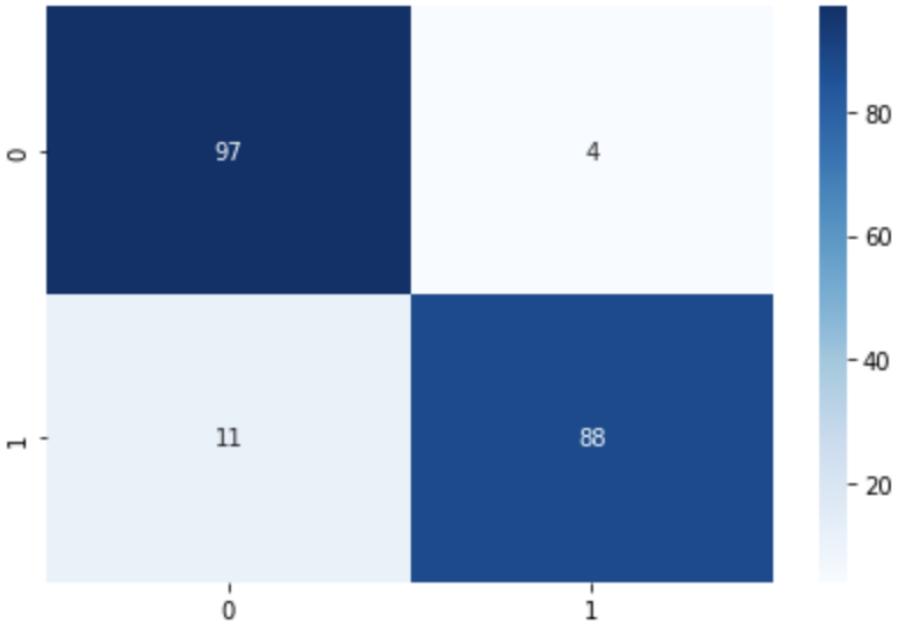
## Best Model

Gradient Boost Classifier gave the best result in all cases with highest accuracy, precission and recall. We are adding the confusion matrix below.

Train Confusion matrix




Test Confusion matrix



# Task 3

## Data analysis & Preprocessing

Data contains 37 columns and 1500 rows. All the columns have all values. Target output class have numerical value. 2 columns contains categorical values. We converted categorical values to numeric value as helps model learn more accurately.

## Model Selection

We tried linear regression, mlp neural network, decision tree, lasso linear regressor, ada boost and gradient boost.

| Model | Train RMSE | Test |
|---|---|---|
| Linear Regression | 664.316 | 655.012 |
| MLP NN | 556.356 | 559.493 |
| Decision Tree | 936.994 | 1031.966 |
| Random Forest | 330.127 | 712.481 |
| Lasso | 665.803 | 653.464 |
| Gradient Boosting | 190.510 | 453.526 |
| Ada Boosting | 589.541 | 650.076 |

## Best Model

Gradient Boost Regressor gave the best result in all cases with highest accuracy and RMSE.