

**Zewail City of Science, Technology and Innovation
University of Science and Technology
School of Computational Sciences and
Artificial Intelligence
CSAI 253 Fall 2024**

Team Member [ID]

Mariam Mohamed Goda[202201223]

Doctor : Mayada Mansour

Classification Problem Report For Credit Card Fraud Detection

[Phase 2 : Kaggle Competition and Model Refinement]

Problem Statement:

The problem that we are trying to solve is that the Credit Card Fraud Detection is an imbalance data set that contains two classes labeled by $\{0,1\}$. 0 represents the non fraudulent transaction while one represents the fraudulent transaction. Our goal is to implement a machine learning model that can deal with the imbalance data which helps in decreasing the error loss and achieve a good prediction from it. As it needs to ensure that the fraud detection is minimized and start to focus on the performance measurement methods for classification to see if you are in progress to our goal or not.

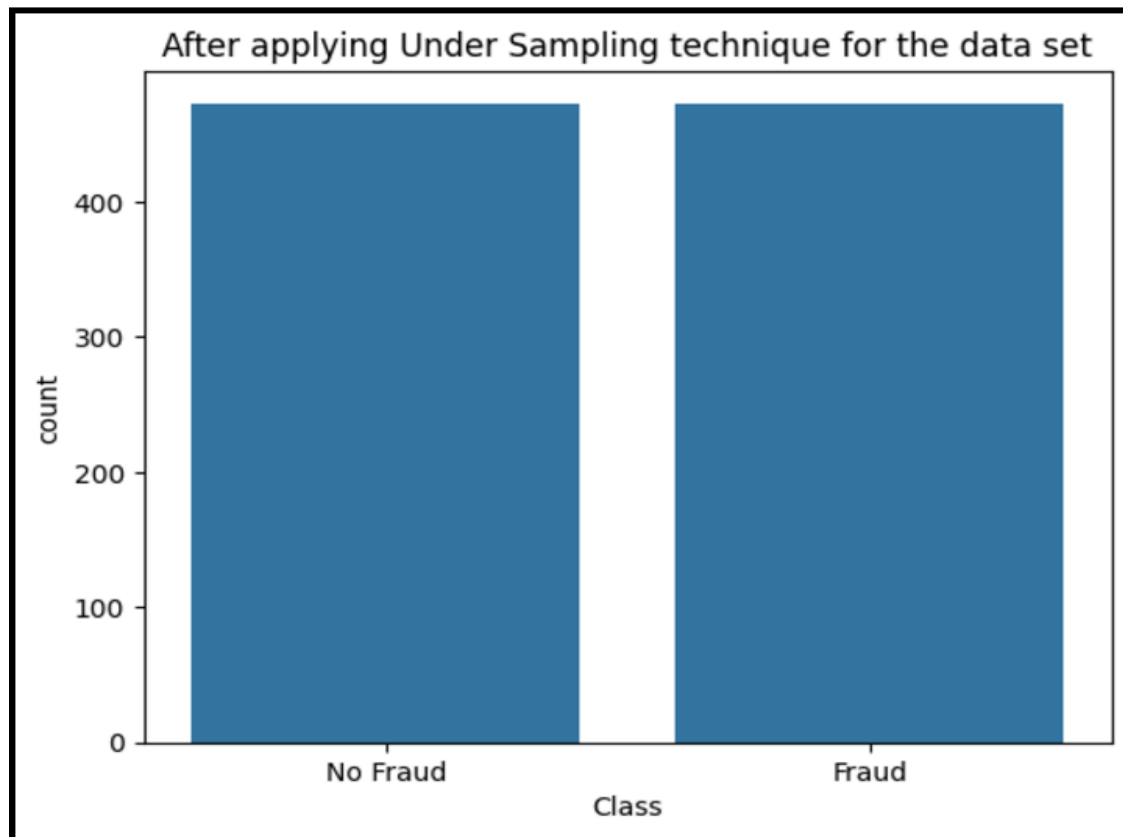
Zewail City of Science, Technology and Innovation
University of Science and Technology
School of Computational Sciences and
Artificial Intelligence
CSAI 253 Fall 2024

Data Exploration and Preprocessing:

Phase one explanation

As shown on the histogram figure, The number of normal transactions equal 283253 while the fraudulent transaction was equal 473 and that will lead to **Overfitting** to the data during the evaluation stage as it will affect the expected outcome due to the high variance. The solution of this stage is to **Resample the data set** before applying the needed models on it. In our project we use the **Under_Sample technique** as it works by taking a sample from the larger class may be equal to the other class value to start work on it with the small class. This will help to decrease the overfitting percentage and make the model learn in an effective way. As Shown in the figure a histogram shows the class col after applying Under sample technique.

Figure (1)



Zewail City of Science, Technology and Innovation
University of Science and Technology
School of Computational Sciences and
Artificial Intelligence
CSAI 253 Fall 2024

This next block shows how we take a sample from the normal transaction class to complete working with it after split the two classes then use the concat function to append them again in the data set with the new normal transaction value and start to implement the models.

```
#start to take a sample from No_Fraud class
No_Fraud_Sample = No_Fraud.sample(n = 473)

#Desgin the new modifed balanced data set
balanced_data = pd.concat([No_Fraud_Sample,Fraud],axis = 0)
balanced_data.head()
```

Phase two new techniques explanation:

According to **data augmentation and synthetic data generation methods** that are needed to be implemented from our searches we got that data augmentation is depended on the training data set that we got to increase the data set size while synthetic data augmentation provide more security protection for the data and works on decrease the data security problem for the data set also has the ability to deal with the imbalance data set . From our searches we found that we can use the **“Smote function”** . It is one from Synthetic Minority Over sampling methods which works on addressing the class imbalance problem by generating synthetic samples for the minority class. It has many extensions that help in enhance the model accuracy and performance as each one can be used for a specific problem statement in the data set for its extensions are {ADASYN , Borderline SMOTE , SMOTE-ENN}. We choose to work using **Borderline SMOTE**. It focuses on defining a synthetic sample next to the decision boundary that is located between

Zewail City of Science, Technology and Innovation
University of Science and Technology
School of Computational Sciences and
Artificial Intelligence
CSAI 253 Fall 2024

the classes. According to the used data set we can implement this method If fraud cases (minority class) lie close to the majority class and are harder to separate them. Also it helps address class overlap by generating samples in critical areas. In addition it helps in reducing the noise by focusing on "hard-to-classify" samples.

This block shows its implementation in the code .

```
smote = BorderlineSMOTE(random_state=42, kind='borderline-1')  
x_train_re, y_train_re = smote.fit_resample(x_train_c, y_train_c)
```

Model Implementation / Selection / Evaluation:

Phase one Models:

As this data needed in the classification target so the implemented model that is going to train and test the data are the Classification models as (Linear classification, Redgie regression, Logistic regression, K-NN Classification) but the best implemented model for The classification models is the Logistic regression then the K-NN classification while the others used more in the regression models. Logistic regression model is a classification algorithm used for the training of binary or multiple classes as it is used on the linear output as it maps output into continuous values between range of [0,1] due to the sigmoid function (logistic function).

```
[ ] #Logistic Regression after using the under sample  
    model = LogisticRegression()  
    model.fit(x_train_c, y_train_c)
```



▼ LogisticRegression ⓘ ?
LogisticRegression()

Zewail City of Science, Technology and Innovation
University of Science and Technology
School of Computational Sciences and
Artificial Intelligence
CSAI 253 Fall 2024

This block describe how to implement the logistic regression using its built in function from the `sklearn.linear_model` , first you should fit the data that you are going to work with which is the training data of the features and the target. Then get the predicted outcome and measure its performance and compare with the real value.

As we know it is the best model that can apply on the classification model is the Logistic regression due to its loss function and the ability to decrease the range of the y_{hate} and make it nearest to the real value range. K-NN Classification is a non parametric algorithm used for the classification algorithm as it has the ability to get the nearest labeled class for the needed data point. the implementation of K-nn classification of the code works on giving a range for the value of data point that is going to get it's nearing labeled class then starts to loop on the given range to fit the training data and get the predicted class.

Phase two Models:

In this phase first we work in the preprocessing stage with the Borderline SMOTE function without an under sample stage so there are some trials with it but I am going to identify it.

Trail : Bagging using the Random Forest Model.

```
[ ] rf_clf = RandomForestClassifier(n_estimators=100, random_state=42)
      rf_clf.fit(x_train_re, y_train_re)
```



RandomForestClassifier



RandomForestClassifier(random_state=42)

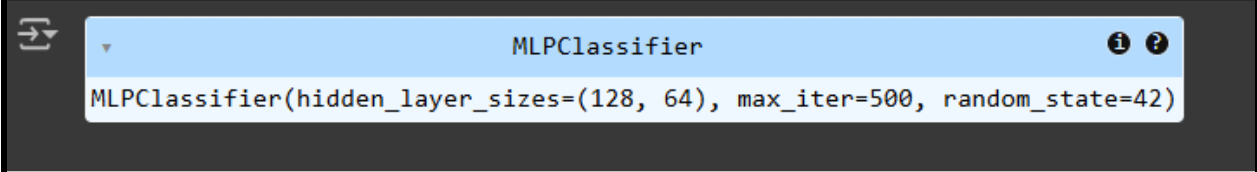
Zewail City of Science, Technology and Innovation
University of Science and Technology
School of Computational Sciences and
Artificial Intelligence
CSAI 253 Fall 2024

In this phase we focus on ensemble learning and the importance of combining the models with each other to get the best performance measurement for our work . we try to use the bagging as it has the ability to train each model in sub data from the training data then combine them. It gives us a good accuracy score which is 0.94 after the submission on kaggle.

Trail :NN With Adam solver.

```
[ ] multiclass_model_3 = MLPClassifier(hidden_layer_sizes=(128, 64),
                                       activation='relu',
                                       solver='adam', max_iter=500,
                                       random_state=42)

multiclass_model_3.fit(x_train_re, y_train_re)
```



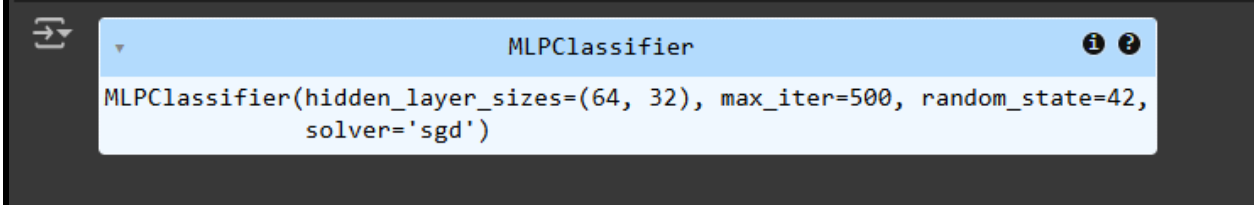
This trial we try to implement the neural network model with RLU as an activation function and Adam optimization method as a solver and give us an accuracy score range between 0.94 and 0.95.

Trail :NN With sgd solver.

This trial we try to implement the neural network model with RLU as an activation function and SGD as a solver and give us an accuracy score range between 0.94 and 0.95.

```
[ ] multiclass_model = MLPClassifier(hidden_layer_sizes=(64, 32),
                                       activation='relu',
                                       solver='sgd', max_iter=500,
                                       random_state=42)

multiclass_model.fit(x_train_re, y_train_re)
```



Zewail City of Science, Technology and Innovation
University of Science and Technology
School of Computational Sciences and
Artificial Intelligence
CSAI 253 Fall 2024

Trail :Voting method

```
from sklearn.ensemble import VotingClassifier

# Create a Voting Classifier
voting_clf_u = VotingClassifier(estimators=[
    ('sgd_clf', sgd_clf), ('dTree_clf', dTree_clf)
], voting='hard')
```

Also we try to use the voting classifier with two models which are decision tree classifier and SGD. From this stage the accuracy score started to be lower than the others try as this score was 0.89.

Trail :Using SVM .

```
[ ] SVM_Class = SVC(kernel='linear',C=10)
SVM_Class.fit(x_train_re, y_train_re)
y_pred = SVM_Class.predict(x_test_c)
print("Classification Report:")
print(classification_report(y_test_c, y_pred))
print(f"Accuracy: {accuracy_score(y_test_c, y_pred):.2f}")
```

Also we try to implement SVM with a linear kernel on the data but also give a lower accuracy score between range of 0.89 and 0.90.

Then we start to use some from our trail with an under sample stage and borderline smote search to make sure that the problem of imbalance data is going to be more solved. I am going to clarify what we got.

**Zewail City of Science, Technology and Innovation
University of Science and Technology
School of Computational Sciences and
Artificial Intelligence
CSAI 253 Fall 2024**

Trail :Using SVM with both preprocessing stages.

We start to see a clear change on accuracy score as it reaches 0.93 score after the submission .

Trail : Voting method .

Try to use the voting method many times as the combination of the models start to improve the accuracy score trail after another trial and the best scores that we reached are :

a) First trial was the same using both the sgd and decision try model which improved the accuracy to be in range from 0.93 to 0.95.

b) The Second trial was the best score that was reached as we got 0.97466 using the voting . In this trial we start to change the hyper parameters for each model as we combine SGD , DECISION TREE , SVM & RANDOM FOREST MODELS after a lot of trials we reach these values for each one as the following block is identified .

```
[ ] #intialize the models that we are going to use(using four models)
sgd_clf = SGDClassifier(max_iter=1000,tol=1e-4)
dTree_clf = DecisionTreeClassifier(max_depth=None,random_state=42)
SVM_Class = SVC(kernel='rbf',C=20,gamma=0.5)
rf_clf = RandomForestClassifier(n_estimators=100, random_state=42)

[ ] voting_clf_2 = VotingClassifier(estimators=[
    ('sgd_clf', sgd_clf), ('dTree_clf', dTree_clf) ,('SVC',SVM_Class),('rfc',rf_clf)
], voting='hard')
```


Zewail City of Science, Technology and Innovation
University of Science and Technology
School of Computational Sciences and
Artificial Intelligence
CSAI 253 Fall 2024

- c) Another trail to the voting but with a lower score which was 0.97032, we start to ask why not if we implement a new model on the last submission to get a higher score but it was wrong. We start to implement the neural network model with the last models to reach what we need as the following block shows.

```
[ ] #initialize the models that we are going to use(using four models)
sgd_clf = SGDClassifier(max_iter=1000,tol=1e-4)
dTree_clf = DecisionTreeClassifier(max_depth=None,random_state=42)
SVM_Class = SVC(kernel='rbf',C=20,gamma=0.5)
rf_clf = RandomForestClassifier(n_estimators=100, random_state=42)
multiclass_model = MLPClassifier(hidden_layer_sizes=(128, 64),
                                activation='relu',
                                solver='adam', max_iter=500,
                                random_state=42)

[ ] voting_clf_3 = VotingClassifier(estimators=[
    ('sgd_clf', sgd_clf), ('dTree_clf', dTree_clf), ('SVC',SVM_Class),('rfc',rf_clf),('nn',multiclass_model)
], voting='hard')
```

-Last three trials we start to use the neural network with the same solvers and activation models but unfortunately the accuracy score starts to be lower again. On the nn trail we start to use the only SMOTE function on the data without any extension but we reach the same value that we got from the last trail. So we started to implement another ensemble learning algorithm to enhance the performance and it was stacking with SVM, decision tree and sgd but the score was lower than the other trials. That was our trials for this data set.

-My conclusion is some models have the ability to work with a specific preprocessing methodology but the others do not. The project helps us in searching for different concepts and new the importance of machine learning models in real life and really was an interesting trail.