**Credit Card Fraud Dataset**

**Data Acquisition Details:**

1. We acquired the dataset from the following site manually

[www.kaggle.com/datasets/mlg-ulb/creditcardfraud?resource=download](http://www.kaggle.com/datasets/mlg-ulb/creditcardfraud?resource=download).

1. The dataset contains transactions made by credit cards in September 2013 by European cardholders.
2. This dataset presents transactions that occurred in two days, where we have 492 frauds out of 284,807 transactions. The dataset is highly unbalanced, the positive class (frauds) account for 0.172% of all transactions.
3. Further processing of the dataset was done using pandas python library.

**Data Cleaning Details :**

1. We inspected the data and found that there was no null or missing value in it.
2. We scaled the Amount and Time data using Robust Scaler as we wanted scaling to be less affected with outliers.
3. Others variables were already scaled and didn’t require any further scaling
4. We divided our data into train and test set so that we can see the performance effectively on unseen data as well.
5. We also used some Oversampling and Undersampling as well on the train dataset so as to see if any performance improvement or not

**Modelling and Evaluation:**

1. Our problem statement was that of binary classification so we used various models from them such as logistic regression, decision tree , KNN and XGBoost
2. We evaluated the performance using ROC-AUC Score since the dataset was highly imbalanced and we were getting very high accuracy.
3. With various models we checked the performance and recorded as below

|  |  |  |
| --- | --- | --- |
| Model | Training Dataset ROC Score | Testing Dataset ROC Score |
| Logistic Regression | 0.976 | 0.950 |
| Decision Tree | 0.883 | 0.851 |
| KNN Classifier | 0.872 | 0.821 |
| XGBoost Classifier | 0.894 | 0.892 |
| Random Forest Classifier | 0.883 | 0.851 |

We further tried to improve the performance using oversampling and undersampling using XGBoost Model but didn’t get much improvement in results with it.

|  |  |
| --- | --- |
| **Method** | **ROC-AUC Score on Test Dataset** |
| XGBoost without Sampling | 0.8927955 |
| XGBoost with RandomUnderSampling | 0.8927955 |
| XGBoost with Oversampling | 0.8927955 |
| XGBoost with Adasyn | 0.8927955 |

Next we performed Hypertuning of XGBoost models and found below performance improvement. We can see that indeed we find good performance improvement with these results.

|  |  |
| --- | --- |
|  | Roc Auc Score |
| XGboost baseline | 0.8927955 |
| XgBoost Hypertuned | 0.9510457 |

**Student Examination Performance Dataset**

**Data Acquisition Details:**

1. We acquired the dataset from the following site manually from following open source dataset,

<https://www.kaggle.com/datasets/spscientist/students-performance-in-exams>

1. The dataset is related to performance of students in various exams and some there informations such as Level of Education, Race, Gender, etc.
2. The data consisted of 1000 rows and we choose Math Score as our target column and formulated the problem as a regression technique.

**Data Cleaning Details :**

1. We inspected the data and found that there was no null or missing value in it.
2. We did the One Hot Encoding for the variables since most of the variables were categorical type.
3. We used Cross validation technique for evaluation the model performance.

**Modelling and Evaluation:**

1. We formulated the problem as Regression and used R-squared score for performance metric using cross-validation technique
2. We used the baseline models of linear regression, Decision Tree Regressor, XGBoost Regressor and KNN Regressor for studying the performance.
3. The performance scores can be seen as below with various models:

|  |  |
| --- | --- |
| **Model** | **Training Dataset R-Squared Score** |
| Linear Regression | 0.2150 |
| Decision Tree Regressor | -0.1422 |
| KNN Regressor | 0.0283 |
| XGBoost Regressor | -0.1342 |

We can see that performance is best using linear regression but other models performance has not been good.

**Hypertuning**

1. We tried to see if there is any improvement using Hypertuning so we selected XGBoost regressor for this.
2. We used grid search approach and selected parameters which can be hypertuned such as number of boosting rounds, learning rate, etc.
3. With the best parameters obtained we again tested the cross-validation performance as shown below:

|  |  |
| --- | --- |
|  | R-Squared Score |
| XGboost baseline | -0.1342 |
| XgBoost Hypertuned | 0.1654 |

1. So we can see the performance improvement using hypertuning method.