**Problem Statement:**

A credit card is one of the most used financial products to make online purchases and payments. Though the Credit cards can be a convenient way to manage your finances, they can also be risky. Credit card fraud is the unauthorized use of someone else's credit card or credit card information to make purchases or withdraw cash.

It is important that credit card companies are able to recognize fraudulent credit card transactions so that customers are not charged for items that they did not purchase.

The dataset contains transactions made by credit cards in September 2013 by European cardholders. 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.

We have to build a classification model to predict whether a transaction is fraudulent or not.

**Project Understanding:**

If you receive a call from your bank informing you about your card expiration and the executive requests sensitive details such as your credit card number, expiry date, and CVV number, it is crucial to exercise caution. Sharing such information over the phone poses a security risk and could potentially grant unauthorized access to your credit card account. It is advisable not to disclose these details unless you can independently verify the authenticity of the call. Instead, consider contacting your bank directly using the official contact information provided on their official website or the back of your credit card to confirm the status of your card and renewal process. Always prioritize security and verify the legitimacy of any request for sensitive information.

While digital transactions in India demonstrated a substantial 51% growth in 2018–2019, the issue of their security persists. Instances of fraudulent activities have surged, notably evidenced by approximately 52,304 reported cases of credit/debit card fraud in the fiscal year 2019 alone. The escalating frequency of banking frauds underscores the urgent necessity to promptly identify these deceptive transactions, offering protection to both consumers and banks whose creditworthiness is jeopardized daily. Machine learning emerges as a crucial tool in detecting fraudulent activities.

In your journey thus far, you have acquired knowledge about various machine learning models. Now, you are poised to understand the selection of an appropriate model for your specific objectives and the rationale behind it. The ability to discern models based on different scenarios is a crucial skill for a data scientist or machine learning engineer. Additionally, fine-tuning your model is paramount to achieving the optimal fit for the given dataset.

By the conclusion of this module, you will gain insights into building a machine learning model capable of identifying fraudulent transactions. The module will also equip you with the skills to, encompassing aspects such as model selection and hyperparameter tuning.

**About the Dataset:**

* The data set has a total of 2,84,807 transactions; out of these, 492 are fraudulent. Since the data set is highly imbalanced, it needs to be handled before model building.
* It is important that credit card companies are able to recognize fraudulent credit card transactions so that customers are not charged for items that they did not purchase.
* The dataset contains transactions made by credit cards in September 2013 by European cardholders.
* 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.
* It contains only numerical input variables which are the result of a PCA transformation. Unfortunately, due to confidentiality issues, we cannot provide the original features and more background information about the data. Features V1, V2, … V28 are the principal components obtained with PCA, the only features which have not been transformed with PCA are 'Time' and 'Amount'. Feature 'Time' contains the seconds elapsed between each transaction and the first transaction in the dataset. The feature 'Amount' is the transaction Amount, this feature can be used for example-dependant cost-sensitive learning. Feature 'Class' is the response variable and it takes value 1 in case of fraud and 0 otherwise.
* Given the class imbalance ratio, I recommend measuring the accuracy using the Area Under the Precision-Recall Curve (AUPRC). Confusion matrix accuracy is not meaningful for unbalanced classification.

**Steps Followed in this project:**

* **Exploratory Data Analysis**

1. *Checking Missing Values*
2. *Outliers Treatment:* The entire dataset undergoes PCA transformation, and it is assumed that outliers have been addressed. Consequently, no outlier treatment is applied to the dataframe, despite the presence of outliers being observed.
3. *Checking Classes Distribution w.*r.t Amount of transaction
4. *Plotting Distribution of Variables for both fraudulent & non-*fradulent transactions
5. *Dropping Time Feature:*  *As it has nothing to do with the outcome*

* **Train Test Split**

1. 80/20 Train Test Split
2. Stratify Split to maintain the proportion of classes as a way to fight against imbalanced dataset

* **Feature Scaling using Robust Scaler**

1. As PCA is already performed on the dataset from V1 to V28 features, we are scaling only Amount field

* **Skewness Treatment**

1. Plotting the distribution of a variable to handle skewness
2. Skewness of a distribution is defined as the lack of symmetry. In a symmetrical distribution, the Mean, Median and Mode are equal. The normal distribution has a skewness of 0. Skewness tell us about distribution of our data.
3. Effects of skewed data: Degrades the model's ability (especially regression based models) to describe typical cases as it has to deal with rare cases on extreme values. ie right skewed data will predict better on data points with lower value as compared to those with higher values.
4. If there is skewness present in the distribution use: Power Transformer package present in the preprocessing library provided by sklearn to make distribution more Gaussian

* **Stratified Cross Validation**

1. When the data is imbalanced or less, it is better to use K-Fold Cross Validation for evaluating the performance when the data set is randomly split into ‘k’ groups.
2. Stratified K-Fold Cross Validation is an extension of K-Fold cross-validation, in which you rearrange the data to ensure that each fold is a good representative of all the strata of the data.
3. When you have a small data set, the computation time will be manageable to test out different hyperparameter combinations. In this scenario, it is advised to use a grid search.
4. However, with large data sets, it is advised to use a randomised search because the sampling will be random and not uniform.

* **Model Building Using Grid Search & Performance Metric**

1. In general, taking the maximum ROC-AUC score may be appropriate if your goal is to select the best model. However, taking the mean ROC-AUC score is generally a more robust and reliable way to compare models, especially if you are using cross-validation to estimate performance.
2. The reason why taking the maximum ROC-AUC score may not be as reliable is that it can be sensitive to small fluctuations in the data or the model, which can lead to overfitting. For example, if you have a small dataset, taking the maximum ROC-AUC score may not be as reliable because the score can vary significantly depending on the particular set of examples that are included in each fold. Similarly, if you have a highly variable model (e.g., a model with high variance), taking the maximum ROC-AUC score may not be as reliable because it may be overly optimistic about the true performance of the model.
3. On the other hand, taking the mean ROC-AUC score can be more robust because it smooths out the variability in the data and the model. By taking the mean of the scores across multiple folds, you are more likely to get a reliable estimate of the true performance of the model. Additionally, taking the mean ROC-AUC score can be useful for model selection, as it allows you to compare models in a more systematic and reliable way.
4. We used Grid Search CV for

Logistic Regression

KNN Classifier

Decision Tree

* **Best Hyper-Parameters for each model**
* **LogisticRegression** {'C': 0.01, 'penalty': 'l2'} =
  + Best Mean ROC-AUC score for val data: 0.9797969874466093
  + Mean precision val score for best C: 0.885478588591554
  + Mean recall val score for best C: 0.6295975017349064
  + Mean f1 val score for best C: 0.7341406860856002
* **KNeighborsClassifier** {'metric': 'manhattan', 'n\_neighbors': 9} =
  + Best Mean ROC-AUC score: 0.9274613536399045
* **DecisionTreeClassifier** {'criterion': 'entropy', '': 3, 'min\_samples\_leaf': 1, 'min\_samples\_split': 2} =
  + Best Mean ROC-AUC score for val data: 0.9337472016466822
  + Mean precision val score for best max\_depth: 0.8480952241800844
  + Mean recall val score for best max\_depth: 0.71578379211967
  + Mean f1 val score for best max\_depth: 0.7752315571186218
* **Evaluation over test set**
* **LogisticRegression** {'C': 0.01, 'penalty': 'l2'} =
  + LogisticRegression ROC-AUC Score on Test Set = 0.9752271441778737
  + LogisticRegression F1-Score on Test Set = 0.5977011494252873
  + LogisticRegression Precision on Test Set = 0.4785276073619632
  + LogisticRegression Recall on Test Set = 0.7959183673469388
* **KNeighborsClassifier** {'metric': 'manhattan', 'n\_neighbors': 9} =
  + KNeighbors Classifier ROC-AUC Score on Test Set = 0.9385655570613163
  + KNeighbors Classifier F1-Score on Test Set = 0.824858757062147
  + KNeighbors Classifier Precision on Test Set = 0.9240506329113924
  + KNeighbors Classifier Recall on Test Set = 0.7448979591836735
* **DecisionTreeClassifier** {'criterion': 'entropy', 'max\_depth': 3, 'min\_samples\_leaf': 1, 'min\_samples\_split': 2} =
  + **Decision Tree Classifier** ROC-AUC Score on Test Set = 0.9314465304973987
  + Decision Tree Classifier F1-Score on Test Set = 0.8200000000000001
  + Decision Tree Classifier Precision on Test Set = 0.803921568627451
  + Decision Tree Classifier Recall on Test Set = 0.8367346938775511
* **Conclusion**

| **Model** | **Parameter** | **ROC-AUC Score** | **F1-Score** | **Precision** | **Recall** |
| --- | --- | --- | --- | --- | --- |
| LogisticRegression | {'C': 0.01, 'penalty': 'l2'} | 0.975227144 | 0.59770115 | 0.47852761 | 0.79591836 |
| KNeighborsClassifier | {'metric': 'manhattan', 'n\_neighbors': 9} | 0.938565557 | 0.82485875 | 0.92405063 | 0.74489795 |
| DecisionTreeClassifier | {'criterion': 'entropy', 'max\_depth': 3, 'min\_samples\_leaf': 1, 'min\_samples\_split': 2} | 0.931446530 | 0.82000000 | 0.80392156 | 0.83673469 |

**According to the above test results, best model for our imbalance dataset is Logistic Regression.**

*P.S. Model Deployment would have been useful if the features were known to us & the end user can get a prediction based on the input. But, here we don’t know what V1 to V28 features are, so I have not done model deployment for this project.*