The primary goal of this work is to develop an accurate credit rating model based on personal transactions of bank clients. The model's goals are to decrease client defaults, reduce the ratio of non-performing loans to performing loans, and evaluate credit institutions' financial sustainability and capacity to provide credit and honor customer engagements. The billion-dollar consumer lending industry's financial firms strive to increase revenues while minimizing risk.

Banking Customer Credit-Score Bracket Classification

Tanvi Mathur
Data Science & Analytics Student

Introduction

1. Overview

Researcher: *Tanvi Mathur*
Project title: *Banking Customer Credit-Score Bracket Classification*
Research Question: *Given a client's bank historic transaction and better details, how can we determine the client's Credit score bracket (Good, Standard Poor)*
Project context: *The goal of this project is to create a machine learning system that will categorize banking clients into various credit-score brackets. Banks may therefore make more informed judgements when accepting loan and credit card applications, lowering risk, and increasing profitability.*

2. Executing your plan

What actions have you identified from the rest of this plan? *There are following actions needed for executing:* • *Understanding the dataset and the research problem.* • *Cleaning the data based on understanding the data and its variables.* • *Doing a complete EDA for the data both for individual variables as well as for correlation between target variable and other variables.* • *Applying Unsupervised learnings PCA will be applied by everyone use of a second supervised learning will be based on induvial choice [K-Means, DBSCAN]* • *Finally, individually everyone will implement their supervised learning models like Random Forest, Neural Network, Decision Tree, SVM, XGBoost.* • *Results of individual implementations are shared, and conclusions are drawn based on them.*
What further information do you need to carry out these actions? *The further studies need to identify the right ways to implement the above steps for which we searched over the internet to find things that are relevant and needs to be implemented.*

The data preparation and implementation of machine learning is done using:

- **Programming Languages:** Python [Some team members have done the cleaning and EDA in R as well.]
- For high performance computation infrastructure, we used Jupyter Notebook as editor with PySpark ML libraries.

Data Preparation and Cleaning:

The data preparation and machine learning implementation are done using **Python** programming language and **PySpark** framework to process large dataset easily. The code editor used for the entire process is **Jupyter Notebook**.

Initial Setup: Firstly, the necessary libraries like pandas and numpy along with others are installed and imported in google collab. The google collab notebook is mounted to the drive in with dataset is present and using the pandas read.csv command the dataset is initialized in a data frame. The name of the first data frame is **Initial_dataset**.

Data Cleaning: The process of analysing the data for finding and correcting erroneous, incomplete, badly structured, or insignificant data in a dataset is known as data cleaning. This is necessary to enhance the quality and dependability of data to utilize it for further modelling purpose. **Data Cleaning** was performed on every column in a sequential order. For each column following steps were performed. ***Data inspection:*** For each column in the dataset missing values, special characters, and frequency distribution of values are checked using commands like "isna()", "nunique()", "value_counts()".

Data Transformation: Based on the inspection the data is processed and cleaned.

Special Characters: Columns Annual_Income, Num_of_Loan, Monthly_Balance, Outstanding_Debt, and Num_of_Delayed_Payment had special character underscore which was replaced by space and the columns were converted to numeric.

Negative Values: Columns Num_of_Loan, Delay_from_due_date, Monthly_Balance and Num_of_Delayed_Payment had negative values which were replaced with zero as these values cannot be negative.

Missing Values: Columns Monthly_Inhand_Salary, Monthly_Balance and Num_of_Delayed_Payment had lots of rows with missing values all these columns were replaced with correct median value by using the group by function and grouping the data based on Customer_ID. Median values is used instead of mean value, as median values do not result to abnormal values due to some extreme values in dataset.

Payment_Behaviour Column: The column had 7600 values as **"! @9#%8"** replacement to which was hard to identify, so these values were dropped from the dataset.

Occupation Column: Some rows had a long underscore instead of the occupation of the customer, on further analysis it was seen that each customer had multiple entries and for some entries values were missing. To solve this problem a SQL database was created and using the unique values of Customer_ID column the underscored rows of occupation were filled back with correct values.

Credit History Age column: This column had values in the form "1years and 2months", to convert it into a single unit that is months and make it a numeric column a user defined function **convert_to_months** was created. The function first checks for null values if there are any, they are replaced with a 0 else the string is split and converted to total months.

Payment_of_Min_Amount: There were column values like "Yes", "No", "NM". Here NM is not a relevant value and since we could not replace it with any other value, we simply dropped this value.

The dataset after all this cleaning process is stored in a new data frame **Clean_dataset** which has **81284 rows** and **15 columns**.

Exploratory Data Analysis:

Exploratory Data Analysis is a process of visually examining datasets to summarise their important characteristics. Prior to modelling, EDA is used to examine the insights into the data's core architecture, extract essential components, detect outliers, and find the relationship between target variable and other variables.

Graphical data analysis: The use to visuals to understand the relation between various variables and to analyse each variable for a better understanding and interpretation.

Statistical summary of numeric data: The table below shows the statistical summary of numeric variables of dataset providing a concise overview of key numerical measures, such as mean, median, maximum, minimum, and standard deviation.

	Annual_Income	Monthly_Inhand_Salary	Num_of_Loan	Delay_from_due_date	Num_of_Delayed_Payment	Outstanding_Debt	Credit_History_Age	Total_EMI_per_month	Monthly_Balance
count	9.240000e+04	92400.000000	92400.000000	92400.000000	92400.000000	92400.000000	92400.000000	92400.000000	92400.000000
mean	1.767569e+05	4199.067540	6.830498	21.081926	29.882446	1425.550230	225.483966	1408.948563	403.473260
std	1.434177e+06	3188.886380	58.410875	14.858830	219.542204	1154.398294	95.970033	8324.648660	214.701250
min	7.005930e+03	303.645417	0.000000	0.000000	0.000000	0.230000	1.000000	0.000000	0.000000
25%	1.943348e+04	1625.558333	1.000000	10.000000	9.000000	566.210000	154.000000	30.291558	270.239722
50%	3.757392e+04	3093.153333	3.000000	18.000000	14.000000	1165.670000	231.000000	69.100853	337.185717
75%	7.284956e+04	5967.991667	5.000000	28.000000	18.000000	1945.030000	292.000000	160.992411	471.979471
max	2.419806e+07	15204.633333	1496.000000	67.000000	4397.000000	4998.070000	404.000000	82331.000000	1602.040519

Figure 1: Statistical Summary Table representing all numeric variables of dataset.

Univariant Analysis: The univariate analysis helps in understanding the distribution of values for a single variable. According to dataset there were 15 variables, and we performed the analysis on all of them, which helped us understand the distribution of target variable and identify if there are any outliers in other variables. Below is the box plot of the variable (Num_of_Loan, Num_of_delayed_payment) with outliers which need to be handled and the target value distribution (Chakraborty, 2021).

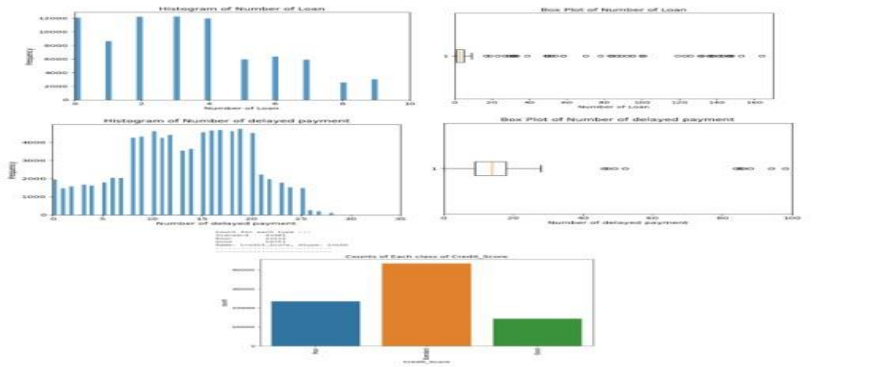
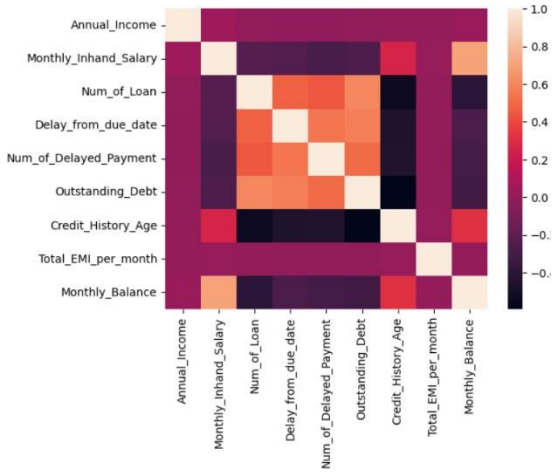


Figure – Representing plot for target variable (Credit_Score) and other variables which need outlier handling



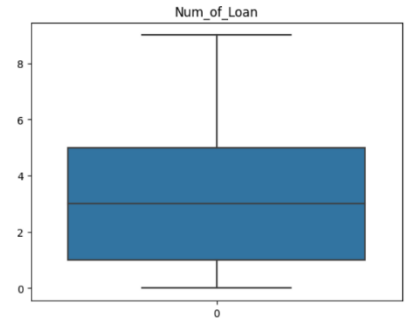
Multivariant Analysis: Multivariate analysis is a statistical approach for identifying patterns and gaining insights by analysing the interactions between numerous variables in a dataset (STEVENS, 2022). According to the dataset we explored the different variables with respect to our target variables.

For instance, a correlation plot is made between all numeric variables to see the correlation of each one of them with one another. The relation of numeric variable is shown using box plot as target variable is categorical while the relation of categorical variable with target variable is shown using contingency

table. For more graphs of univariant and multivariant analysis please refer to the code submitted in the appendix section of wiseflow.

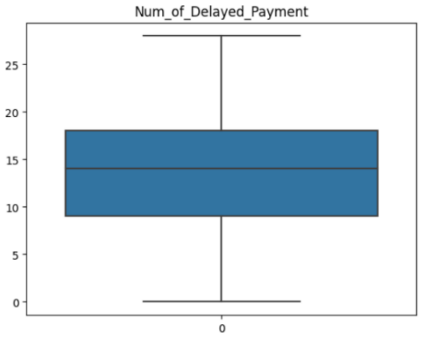
Outlier detection and handling: Outliers are defined as points that deviates from other points this may be due to errors or wrong value imputation. Handling outliers is important as outliers can affect the results of modelling.

Number of Loan Column: Using the box plot in the graphical analysis it was identified that there were lot of outliers in this column. Further, it was seen there were very high values in number of loans which is not possible, so we replaced all the values greater than 9 with the median value of loan based on Customer_ID.



Number of Delayed Payment Column:

Using the box plot in the graphical analysis it was identified that there were lot of outliers in this column. Further, it was seen there were very high values in number of delayed payments which is not possible since only till few chances its acceptable in real scenario, so we replaced all the values greater than 28 with the median value of delayed payment based on Customer_ID.



After dealing with the outliers, the dataset is saved in a csv file before any further processing.

This file is used to create the **encoded_dataset** by utilising feature extraction.

Feature Extraction: We cannot input textual data to any machine learning algorithm since the machine learning algorithm does not understand text data. It solely interprets numerical data. Feature Extraction from Text is the process of transforming text input into numbers (Shankar, 2022).There are 4 column in our dataset that are categorical and needs to be converted to be analysed by algorithm, one of these columns is our target variable.

Occupation Column: The column is categorical and have no natural order this is the reason that we used **One Hot Encoding** to convert it into numeric. Since this encoding technique is specifically used when the variables are not having a natural ordering or hierarchy.

Payment of minimum amount Column: This column had values like yes and no which are considered to have a natural order and hence **Ordinal Encoding** which is used for data with natural ordering or hierarchy is applied to convert this column into numeric.

Payment behaviour Column: This column had values which had a natural order like low_spent_small_value, high_spent_medium_value and others hence **Ordinal Encoding** which is used for data with natural ordering or hierarchy is applied to convert this column into numeric.

Credit score Column: This is the target variable of dataset which has categorical values like poor, standard, good which is again considered to be having a natural order and hence we used **Ordinal**

Encoding to convert into numeric to be understandable by the machine learning algorithm.

Finally, after all the processing this **encoded_dataset** is stored in a csv file and a copy is downloaded to the local system.

Balancing Target Variable:

Initially the "Credit_Score" variable has three classifications namely poor standard and good which were converted to ordinal numeric values as 0,1 and 2 respectively. The imbalance was high as shown in below table.

Value	Imbalanced Count
Poor (0)	26773
Standard (1)	49182
Good (2)	16445

This is highly imbalanced data and might impact the results of machine learning model. Thus, to prevent this we apply **SMOTE** which is Synthetic Minority Oversampling Technique which helps in adding the synthetic samples of minority class in between existing minority class samples, this helped in getting evenly distributed target variable (Brownlee, 2021). It is applied to only training dataset and not to the test dataset. It's clear from below table that the train dataset target variable is balanced.

Value	Balanced Train dataset Count
Poor (0)	34487
Standard (1)	34487
Good (2)	34487

Feature Scaling:

Feature scaling is the process of normalising the range of features in a dataset, it can be done using either normalization or standardization. In machine learning contexts, standardisation is favoured over normalisation since it is especially significant when evaluating the similarities between features based on specific distance measurements. This is especially noticeable in Principal Component Analysis (PCA), a dimensionality reduction approach (Jason, 2020). Scaling is applied just to the train dataset as if we apply scaling to both the training and test datasets, we risk leaking information from the test dataset into the training dataset. This can result in overfitting, where the model becomes too closely tailored to the training dataset and does not generalize well to new data.

Unsupervised Algorithm-1: Principal Component Analysis [PCA]:

PCA is an unsupervised learning approach for prominent dimensionality reduction that seeks to transform a high-dimensional dataset into a lower-dimensional space while maintaining most of the information. PCA works by determining the directions in the data that represent the most variance and projecting the data onto those directions, which are referred to as principal components (Ajitesh, 2023).

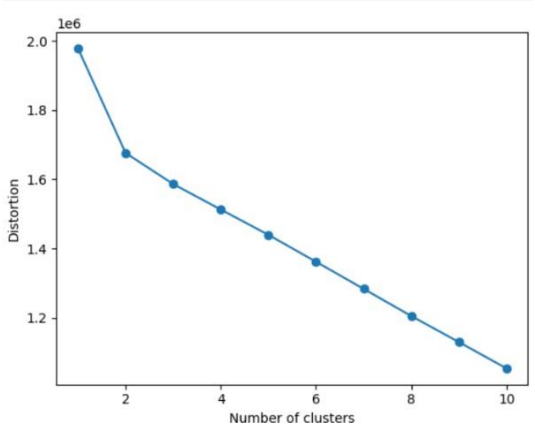
Steps for implementation of PCA:

- Firstly, we specified the amount of variance that should be preserved while applying PCA. The explained variance parameter defines how much of the source data's total variance should be kept in the converted data. In our dataset we applied 0.85 as the variance to be restored.
- Then PCA fit and transform are applied to the scaled train and test features dataset. This resulted in dimensionality reduction as features reduced to 17 from 25.
- Finally, a new dataset for both train and test are created with this reduced features and target variable.

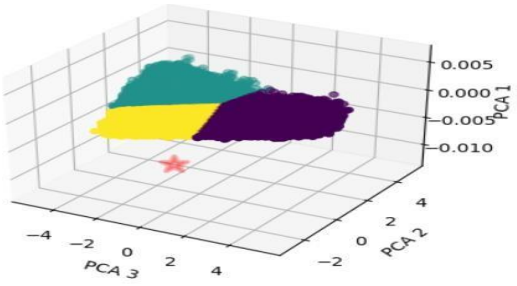
Unsupervised Algorithm-2: K means Clustering Algorithm:

This is an unsupervised clustering technique that separates the provided data into “K” groups. Each cluster has a cluster canter, also known as the centroid. Clustering is a vector quantization approach based on signal processing that attempts to divide 'n' number of observations into 'k' clusters, with each observation belonging to the cluster with the closest mean, i.e. (cluster centres or cluster centroid), which serves as the archetype of the cluster.

Steps to implement K-Means Clustering:



- Installation of the required packages.
- Using the Elbow method on the pca_X_train dataset to build an Elbow plot displayed on the side. The clustering is performed on a range of “K” values for instance 1 to10 and an overall grade is assigned to all clusters for each value of “K”.
- Based on the Elbow plot best K value is opted for clustering, which is **“3”** in case of our dataset since the target variable has three values and the Silhouette score is 0.1417894417165213.
- The cluster are formed based on the K value for both train and test datasets and are added to the datasets. Further, the Silhouette score is calculated for the train data that is pca_X_train and the train_cluster_labels. Three clusters are formed as K value was three.
- Since even after applying PCA the number of features were high, a heatmap was plotted to identify features which were most correlated with the target variable (Credit_Score).



- Finally, the K-Mean cluster was plotted including only the features which showed high correlation in the heatmap with the target variable.
- The K-Means applied train and test dataset were stored in two data frames for future modelling.

Machine Learning Prediction:

The machine learning algorithm chosen was **Random Forest**. Random Forest is a supervised learning method. The excellent accuracy, resilience, feature significance, adaptability, and scalability of random forest makes it a popular technique for classification and regression applications. Random Forest decreases overfitting and is less susceptible to noise and outliers in the data (R, 2023).

Data Preparation: The data was prepared for random forest modelling by applying missing values treatment, categorical data conversion to numeric, standardization and outlier identification. The highlight was that the target variable in the dataset was not balanced and had 3 values [Poor, Standard, Good], so balancing was applied using SMOTE and then the balanced dataset was formed.

Different Datasets were used to performed random forest modelling:

1. Balanced Dataset.
2. Balanced PCA applied Dataset.
3. Balanced K-Means applied Dataset.

Random Forest Training: The three datasets on which we will apply modelling are first trained through following steps.

- Firstly, all these 3 datasets had a train and test set for which the features and target variables are separated.
- Then assembler is used on the train and test datasets as it helps integrate several input characteristics into a single vector column. It is a stage of transformation that aids in the preparation of data for use in training a machine learning model.
- Then random forest classifier model is initialized with parameters like number of trees and maximum depth.
- Finally, the initialized model is fitted into the train dataset.

Random Forest Prediction:

- After model is trained, it goes for prediction which is done on the test dataset.
- The transform method is used to apply the model on test dataset to generate the predictions.
- Then evaluator method is applied on the predictions generated and test accuracy of the model is obtained.

Random Forest Accuracy: There were three different test accuracies obtained from 3 different datasets which are mentioned in the table below. The below results are for **numTrees=100, maxDepth=10**.

Dataset Name	Test Accuracy	Precision	Recall	F1 score
[*balanced_train_dataset, balanced_test_dataset*]	0.635	0.697	0.630	0.629
[*pca_train_dataset, pca_test_dataset*]	0.588	0.640	0.588	0.589
[*cluster_train_dataset, cluster_test_dataset*]	0.587	0.639	0.587	0.589

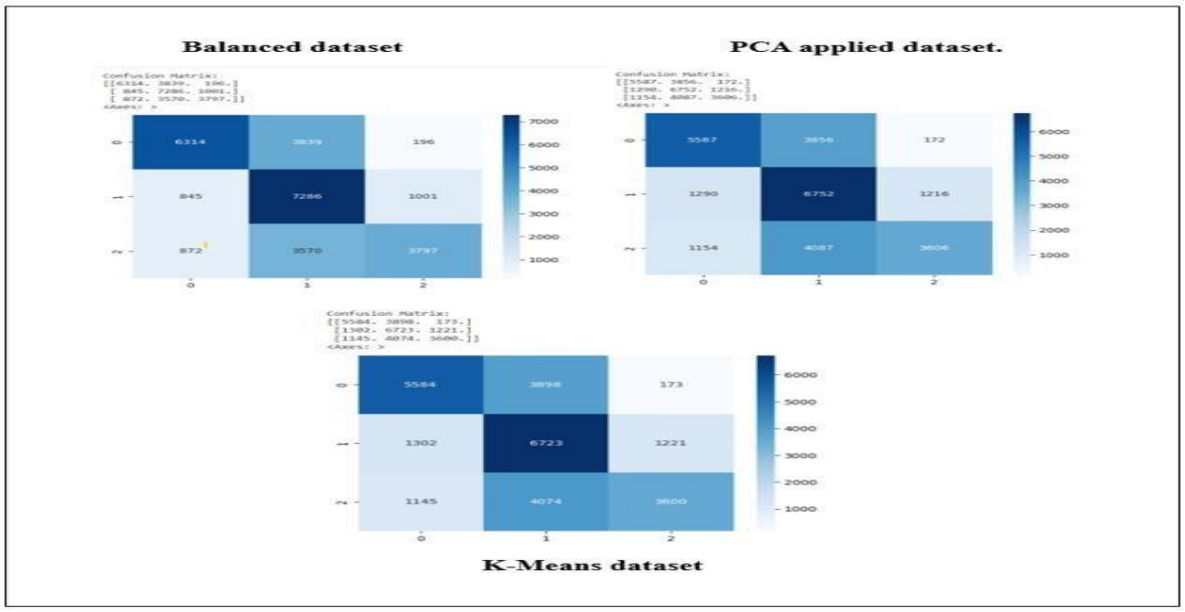


Figure – The Confusion Matrix & Heatmap for all modelling applied datasets.

Based on the Test Accuracy of all the three datasets, **Balanced Dataset** with no PCA or K-Means implementation gave the best score. Therefore, now hyper tunning is applied on the balanced dataset.

Hyperparameter Tuning: The process of determining the ideal hyperparameters for a machine learning model that results in the best performance is referred to as hyperparameter tuning. In case of random forest parameters like number of trees and depth are set before the modelling process.

First, the train balance dataset is divided into train and validation set with 80:20 ratio and then then the train, validation and test dataset are transformed using assembler. Then the random forest classifier is initialized and defining pipeline with the random forest classifier. Pipeline is a method of organising and automating machine learning process activities. It enables you to connect numerous stages, each of which can execute a different action on the data.

The parameters for the tunning are then assigned in the param grid with number of trees as 90 and 110 as we already saw the performance of data with 100 trees and depth as 5 and 15 as we saw the performance with depth 10 earlier. The reason we included the combination of two is that model training was getting crashed with more values due to high volume of data and system incompetency. Even execution of these values in Spark took close to **40 min**. Cross validation is the next important step it splits the data into further K folds and run the full K-Fold CV procedure numerous times, each time with a different model setup. Then we compare all the models, choose the best one, train it on the entire training set, and assess it on the testing set.

The results obtained after hyperparameter tunning are mentioned in below table.

Dataset Name	**Test Accuracy**	**Precision**	**Recall**	**F1 score**	**Validation Accuracy**
[*hyper_train_dataset, hyper_test_dataset*]	0.664	0.711	0.665	0.666	0.753

The best parameters from hyperparameter tunning are given below:

Parameters	**Best Results**
Number of Trees [numTrees]	**110**
Maximum Depth [maxDepth]	**15**
Best Validation Accuracy with above parameters	**0.7543 (75.43%)**
Test Accuracy	**0.664 (66.4%)**

The best accuracy shows that the model correctly categorises **75.43%** of the cases in the validation dataset, but the test accuracy was **66.4%** for the dataset. This value shows the model's overall performance and may be used to compare it to other models or as a starting point for improvement.

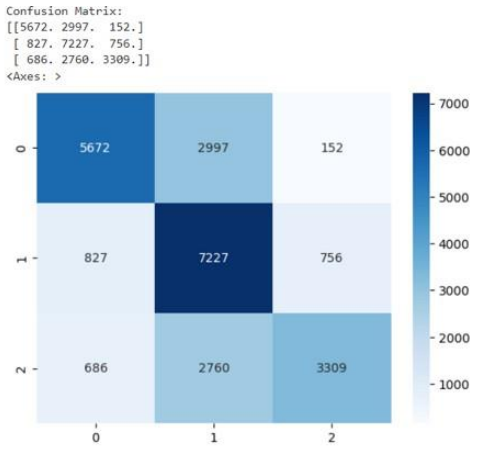


Figure – showing the confusion matrix & heatmap for hyperparameter tunning.

The generated heatmap depicts the confusion matrix visually, with darker colours signifying higher values and lighter colours suggesting lower values. This makes it possible to spot any trends or imbalances quickly and easily in the classification model's performance.

High-Performance Computational Implementation:

High-performance computing is the implementation of parallel data processing to boost computer speed and execute difficult processes. This is achieved by pooling computing resources so that even complex systems may respond to user demands and requirements efficiently, smoothly, and swiftly. As a result, it provides much more power and productivity than typical PCs, laptops, and hosts (Awati , 2021). The dataset is combined with Pyspark and use the Random Forest approach to evaluate the findings**.** In terms of Pyspark, it is now one of the most popular and widely used big data analytics solutions as compared to Hadoop, yet Hadoop is still widely used by many enterprises. Since it transmits information to disc during intermediate processing, Hadoop MapReduce is frequently slower than Pyspark. As the pyspark apps run in their own thread, with Spark Session objects in the driver programme serving as the coordinator. The resource managers provide jobs to workers, which are often assignments for a single split. A job performs a series of operations on the dataset and its division, resulting in fresh split data. Because repeating algorithms operate on the dataset often, it is desirable to store them between repeats. The results are either returned to the driver program or stored in the disk (Haitan , 2020).

We have followed the below steps to achieve results using the Pyspark:

Importing required libraries: Spark Session and Spark Conf. are imported, and Spark Session and Context are formed.

Creating a Spark session: A spark session is created using object name **"spark"** and context object is set using **"sc".** A cluster manager is set which in the case of the implementation is a **"local [*]"**, which means application runs locally and app name is created which is **"New"** for our code. Finally, using **"getOrCreate"** a new session is created.

Loading the dataset: The train and test dataset we are using are imported in spark using **spark.read.csv** function and the schema of train dataset is analysed.

Prepare the train and test dataset: The datasets are distributed into train and test sets to perform machine learning techniques like Random Forest, SVM and many others.

Machine Learning: For our analysis we had 3 different train and test datasets to apply Machine learning model on them.

Balanced Dataset: The dataset that was encoded for all category variables and target variable values that were not balanced has been balanced, and it is now utilised for modelling.

Balanced PCA applied Dataset: PCA was applied to the balanced dataset and features reduced from 25 to 17 based on 0.85 variance. The train and test dataset were formed for these and is now utilised for modelling.

Balanced K-Means applied Dataset: Moving on K-Means clustering was applied on the PCA reduced dataset and a cluster column was added to the dataset so now this new dataset is also utilized for modelling.

These three different models are utilized to perform **Random Forest Classification using Spark**. The results for these are compared and the one with best outcomes is further hyper tuned for checking if better results can be obtained.

Performance Evaluation and Comparison of Methods:

1. Performance Evaluation of **Random Forest Classifier**: The best model after performing hyper tunning are as follows.

Parameters	**Best Results**
Number of Trees [numTrees]	**110**
Maximum Depth [maxDepth]	**15**
Best Accuracy with above parameters	**0.7543 (75.43%)**
Precision	**0.711**
Recall	**0.665**
F1 score	**0.666**
Test Accuracy	**0.664**

The results of hyperparameter tunning indicates how consistent and dependable the dataset is and that **66.4%** cases are categorized correctly in the dataset using Random Forest model.

Performance Evaluation of other machine learning models:

Model Used	Dataset Used	Accuracy	Precision	Recall	F1 Score
Decision Tree	Balanced	0.65	0.65	0.65	0.65
Decision Tree	Imbalanced	0.63	0.67	0.63	0.64
SVM	MultiClass	0.645	0.656	0.642	0.615
SVM	Binary	0.796	0.77	0.823	0.796
XGBoost		0.726	0.708	0.703	0.705
TabNet Original	Original dataset	0.57	0.68	0.6	0.6
TabNet Transformed	PCA applied optimized dataset	0.6	0.61	0.57	0.57

The above table represents the results by using same dataset in different scenarios. The results show that the SVM binary classification gives best accuracy of **79.6%.** The XG Boost performs best for multivariant classification with accuracy of **72.6%** while neural network performed on original dataset gives the least accuracy of **57%**.

Discussion of the findings:

The Random Forest gave validation accuracy of **75.43%** while the test accuracy was **66.4%**. Since the model is giving better results during training it can be due to overfitting which can be due to features selected for the data. Therefore, a method which is more sensitive while selecting features could be ideal for this dataset. Looking at the other models the results suggest that decision tree is better suited for datasets where false positives are costly due to its higher precision, but its recall score remains the same. Therefore, the model should depend on the problem and the relative importance of precision and recall. It can also be seen from SVM binary model that results are more accurate for binary target variable