Comparative Study of Machine Learning Models for Predicting Term Deposit Subscriptions

Ayobami Agboola (15620073)
MSc Data Science and Computational Intelligence
Faculty of Engineering, Environment and Computing,
Coventry University,
Coventry, United Kingdom

***Abstract* —The prediction of term deposit subscriptions is critical task in the banking sector, enabling targeted marketing and improving resource allocation. In this study, a publicly available bank marketing dataset was analysed to develop four machine learning predictive models, random forest, logistic regression, support vector machines (SVM), and a stacked ensemble combining random forest and logistic regression. Each model was evaluated both with and without hyperparameter optimization, using metrics such as accuracy, precision, recall, F1-score, and ROC-AUC. The results indicate that random forest and the stacked ensemble models outperformed logistic regression and SVM, achieving the highest ROC-AUC scores of 0.91. Random forest demonstrated superior accuracy (90%) and precision (81%), while the stacked ensemble model show a robust balance between precision (69%) and recall (82%). Logistic regression and SVM also excelled in recall, emphasizing their utility for minimizing false negatives. Feature importance analysis for all the models revealed "duration," as a critical predictor, "previous contact outcome", "balance", "job type", etc were also identified as predictors, which emphasized the need for feature-specific optimization in future campaigns. This research highlights the efficacy of ensemble methods and the critical role of feature engineering in addressing imbalanced classification challenges in the banking domain.**

Keywords—Term deposit, random forest, logistic regression, SVM, stacked ensemble

I. INTRODUCTION

Marketing is a process whereby products are presented in appealing and attractive ways to clients and customers to serve their requirements. There are two ways of marketing, direct marketing and mass marketing [1]. Direct marketing plays a crucial role in the banking sector, serving as a strategic conduit for personalized and targeted communication with clients. Banks can optimize their resource allocation, personalize customer engagement, and enhance conversion rates by identifying potential subscribers [2]. Thus, the accurate forecasting of customers' behaviour is crucial for improved marketing efficiency, enhanced customer segmentation, and optimized decision-making. Machine learning techniques have rapidly evolved the predictive analytic landscape over the years, and financial institutions can leverage these techniques for improved decision making process, particularly in predicting customer behaviours [3]. This study aims to investigate the predictive performance of four prominent classification models, random forest, support vector machine (SVM), logistic regression, and an ensemble model combining Random Forest and Logistic Regression on a real-world bank marketing dataset, to assess how the implemented models can enhance the effectiveness of direct marketing campaigns.

II. RELATED RESEARCH

The prediction of term deposit subscriptions has become a focal area for financial institutions seeking to optimize marketing campaigns, reduce costs, and improve customer targeting. Various studies have explored machine learning approaches to enhance predictive accuracy in this domain. In a study, [4] explored the same real-world dataset used in this study to predict term deposit subscriptions. They employed classification techniques like decision trees, logistic regression, neural network, and SVM, achieving significant results. Their study underscored the importance of feature engineering and pre-processing in improving model accuracy. Reference [5] tackled the issue of class imbalance in marketing datasets, proposing that under-sampling can lead to improved prediction accuracy. [6] explored an ensemble learning method for predicting term deposit subscriptions. The results demonstrated that classifiers significantly improved predictive performance with a balanced dataset. Reference [7] highlighted the critical role of hyperparameter optimization in improving the performance of machine learning models. Their study emphasized grid search and randomized search for optimizing algorithms like SVM and Random Forest, showing substantial gains in accuracy which aligns with our use of hyperparameter-tuned models to enhance predictive accuracy for the term deposit dataset.

The reviewed studies collectively highlight the importance of robust preprocessing, class imbalance handling, and hyperparameter tuning in predictive modelling for financial datasets. The findings underscore the relevance of our selected methods in predicting term deposit subscriptions.

III. PROBLEM AND DATASET

Problem Description and Significance

In the domain of banking and finance, predicting whether a customer will subscribe to a term deposit following a marketing campaign is a classification problem of high practical significance and the ability to accurately identify potential subscribers can lead to more efficient resource allocation in marketing campaigns, reduce operational costs while maximizing returns. This study implements and compares four different predictive models to identify potential term deposit subscribers based on the binary target variable. The problem is more challenging due to the class imbalance nature of the target variable, in which a far greater number of customers declined the offer compared to those who accepted it. Additionally, the dataset comprises both categorical and numerical features, necessitating careful preprocessing and feature engineering to ensure optimal model performance. Through rigorous hyperparameter tuning, feature engineering, and evaluation, we aim to determine the most effective methods for addressing this practical and impactful classification problem. The result would provide decision-makers with actionable insights which will enable prioritizing high-probability customers and possibly design personalized marketing strategies.

Dataset Description

A dataset on bank telemarketing taken from the UCI machine learning repository was used in this study. The data was a result of direct marketing campaigns based on phone calls by a Portuguese banking institution from May 2008 to November 2010. The original dataset contains about 45211 instances and 21 attributes, later reduced to 17 attributes [4]. However, the copy of the dataset used in this study contains about 10% of the instances randomly selected from the original dataset of 17 attributes. The dataset has no missing values, includes a combination of categorical and integer input variables representing various elements of marketing campaigns and client demographics. The target variable (y) of the dataset indicates whether a customer subscribes to a term deposit. However, the dataset exhibits class imbalance, with a significantly smaller proportion of positive responses (yes) compared to negative ones (no).

IV. METHODOLOGY

Our goal is to build and compare predictive models, including Logistic Regression, Support Vector Machines (SVM), Random Forest, and an ensemble of Random Forest and Logistic Regression to identify potential term deposit subscribers.

Random Forest

Random forest classifier is composed of a pre-defined number of binary decision trees where each tree is generated using a random vector sampled independently, and each tree casts a unit vote for the most popular class to classify an input vector [8]. It is an ensemble learning technique, well-suited for handling feature interactions and is robust in managing imbalanced datasets via class weighting and resampling techniques.

Logistic Regression

Logistic Regression is a statistical approach used to analyse methods of predicting a data value that is entirely based on the observations [9]. It is used to explain the probability of a binary outcome with a dichotomous dependent variable (0 or 1) to transform the linear combination of predictor variables into a probability using the logistic (sigmoid) function [10]. The logistic regression model is renowned for its interpretability and simplicity.

Support Vector Machine

Support vector machine (SVM) is based on statistical learning theory, a concept of classifying data by hyperplane separation in the N-dimensional space. Hyperplanes can be defined as decision boundaries that help to classify the data points [11]. SVMs have the aim of determining the location of decision boundaries that produce the optimal separation of classes. In a two-class pattern recognition problem where features are linearly separable, the SVMs select the one linear decision boundary that leaves the greatest margin between the two classes. The margin is the sum of the distances to the hyperplane from the closest points of the two classes [12]. If the features of data are not linearly separable, an SVM algorithm can pre-process the data and represent the features in a higher-dimensional space in which they can become linearly separable [13]. SVM was used because of its ability to define complex decision boundaries and the linear kernel was applied to capture the linear relationships in the data.

Ensemble Model

Ensemble learning (EL), an established ML technique, stands out as a robust approach by amalgamating predictions from multiple models to enhance overall performance and predictive accuracy [14]. EL models are meta-models that develop models by exploiting multiple weak classifiers and integrating obtained results to achieve stronger classifiers. Stacked ensemble ML model typically comprising heterogeneous models generates the final prediction by combining multiple strong models and aggregating their results. In the first level, stacking models consist of several base models, while in the second level, a meta-model is created, taking into account the outputs of the base models as input [15]. The logistic regression ensemble with Random Forest aims to synergize their complementary strengths.

V. EXPERIMENTAL SETUP

Exploratory Data Analysis

The obtained data was explored before being split into training and test batches for the models. The first crucial step was loading the necessary libraries as well as the models' data files, such as numpy, pandas, seaborn, and utilizing the scikit package on Python. The dataset.info() revealed 4521 entries, 17 columns, where 10 columns were of dtype "object" and the others are of dtype "int64". There was no null value in any of the entries. The pairplot and countplot methods of seaborn were employed to respectively visualize the relationship between the numerical columns (“Fig. 1”) and number of customers who subscribed for the term deposit (“Fig. 2”). The relationship between the numerical features was also examined using a correlation matrix (“Fig. 3”), where only pdays (number of days past before the current contact) and previous (number of previous contacts before current contact) show a week correlation (0.58) and no correlation exist between other features.

Model Development

The study explores two dimensions for each model, the baseline performance without hyperparameter optimization and enhanced performance following hyperparameter tuning. The hyperparameter optimization process employs grid search with cross-validation, ensuring rigorous model evaluation and parameter selection. Metrics such as precision, recall, F1-score, and ROC-AUC are employed to assess the effectiveness of the models in classifying both classes, particularly the minority class. Additionally, feature importance analysis is conducted for interpretability, providing insights into key drivers of customer decisions.

VI. Results

Evaluation

The final performance of each model is given in Table I. The table show that random forest achieved the highest accuracy (90%) and ROC-AUC score (91%), indicating a strong discriminatory power between the classes. However, its recall (58%) and f1-score (60% - 61%) were relatively lower, indicating its challenges in correctly identifying the positive class, which reflects the influence of the dataset’s class distribution imbalance, where the negative instances dominate.

The logistic regression performed well in terms of recall (81% - 82%) and f1-score (72%) particularly excelling at identifying the positive cases. However, its overall accuracy and precision were lower than that of the random forest, highlighting it is also sensitive to class imbalance but reduced precision for false positive control.

The SVM demonstrated comparable recall (81%) and precision (67%) with slightly lower accuracy (82% - 83%) and f1-score (70% to 79). The SVM is effective in separating classes but was less robust in comparison to the ensemble methods.

The stacked ensemble model offered a balance performance, with accuracy (85%), recall (81% - 82%), and f1-score (72%) matching.

The hyperparameter optimization generally improved precision and f1-score for random forest but had minimal impact on other models. This suggests that the models are already operating near their optimal configurations.

The evaluation of model performance is further supported by visual analyses using the confusion matrices (“Fig 4”) and Receiver Operating Characteristic (ROC) curves (“Fig 5”). These visuals provided critical insights into the models’ classification behaviour and their ability to handle class imbalances.

TABLE I. COMPARISON OF APPIED CLASSIFER MODELS

Models	Accuracy	Precision	Recall	F1-Score	ROC-AUC
Random Forest	90%	81%	58%	61%	91%
Random Forest (Tuned)	90%	80%	58%	60%	91%
Logistic Regression	84%	68%	81%	72%	90%
Logistic Regression (Tuned)	84%	68%	82%	72%	90%
SVM	83%	67%	81%	70%	89%
SVM (Tuned)	82%	67%	81%	79%	89%
Stacked Ensemble	85%	69%	81%	72%	91%
Stacked Ensemble (Tuned)	85%	69%	82%	72%	91%

Important features

The analysis of feature importance across the models highlights key variables contributing to the prediction of term deposit subscriptions (“Fig 6”). Random forest top features were ranked by their importance scores in the model, while linear regression and SVM top features were ranked by their absolute coefficients values.

Conclusion

The study highlights the effectiveness of ensemble methods and hyperparameter tuning for predictive modelling in imbalanced datasets. Random forest emerged as the most accurate model with the best ROC-AUC score, while logistic regression and stacked ensembles demonstrated strong recall and F1-Score, making them reliable for identifying positive instances. SVM was competitive but slightly underperformed compared to the ensemble methods. This investigation not only benchmarks the predictive power of these models on the bank marketing dataset but also highlights the trade-offs between interpretability, computational complexity, and performance. The findings contribute to the growing body of knowledge on machine learning applications in finance and offer practical recommendations for implementing predictive models in similar real-world scenarios. Future work could explore techniques such as cost-sensitive learning or hybrid models to further optimize recall and precision, ensuring robust handling of class imbalances in real-world applications.

REFERENCES

[1] V. Abraham and J. Joseph, “An empirical study on direct marketing as the most effective form of marketing in the digitalized marketing environment,” *International Journal of Research Science and Management*, vol. 6, no. 1, pp. 18–24, 2019.

[2] A. Zaki, Nima Khodadadi, Wei Hong Lim, and S. K. Towfek, “Predictive Analytics and Machine Learning in Direct Marketing for Anticipating Bank Term Deposit Subscriptions,” *American Journal of Business and Operations Research*, vol. 11, no. 1, pp. 79–88, Jan. 2024,

[3] S. T. Boppiniti, “Machine Learning for Predictive Analytics: Enhancing Data-Driven Decision-Making Across Industries.,” *International Journal of Sustainable Development in Computing Science*, vol. 1, no. 3, 2019.

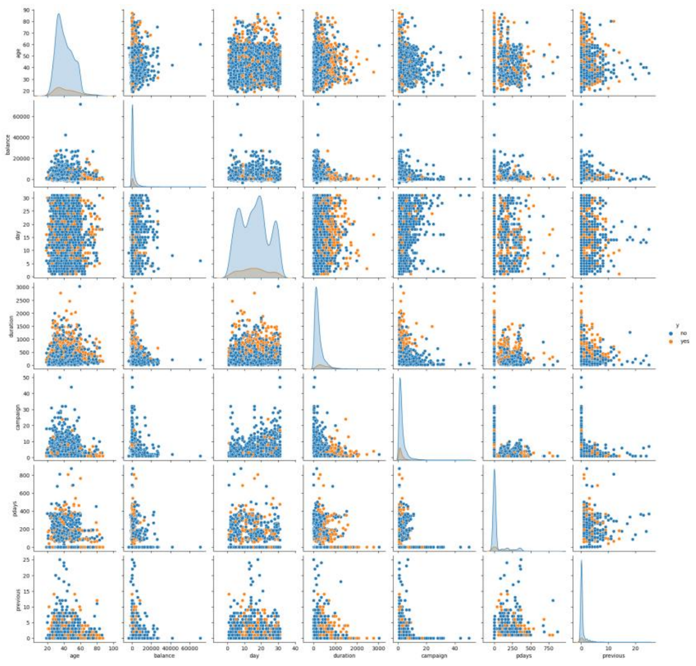


Fig 1. Pair-wise relationship between numerical features.

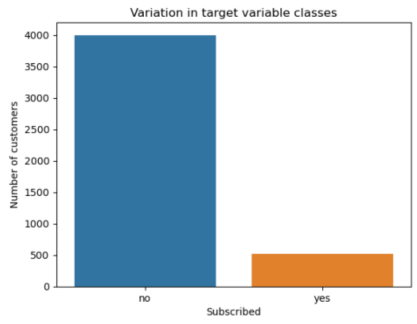


Fig 2. Number of customers who subscribed.

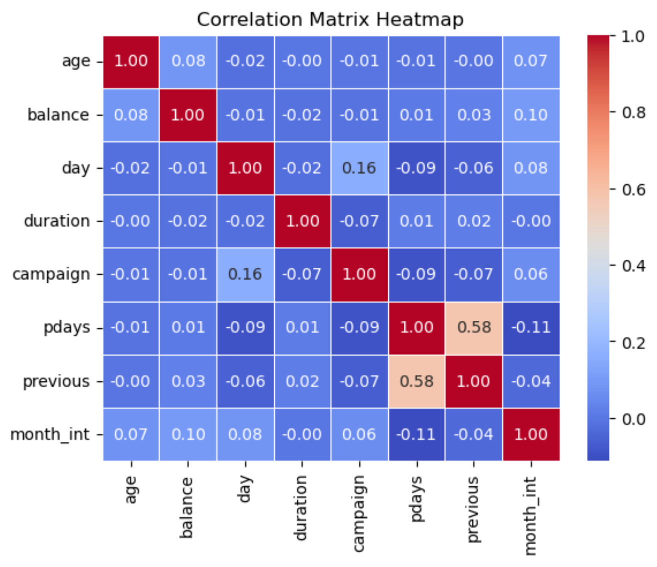


Fig 3. Correlation matrix of numerical features

[4] S. Moro, P. Cortez, and P. Rita, "A data-driven approach to predict the success of bank telemarketing," *Decision Support Systems*, vol. 62, pp. 22–31, 2014.

[5] J. Burez and D. Van den Poel, "Handling class imbalance in customer churn prediction," *Expert Systems with Applications*, vol. 36, no. 3, pp. 4626–4636, Apr. 2009, doi: https://doi.org/10.1016/j.eswa.2008.05.027.

[6] O. Apampa, "Evaluation of Classification and Ensemble Algorithms for Bank Customer Marketing Response Prediction," *Journal of International Technology and Information Management*, vol. 25, no. 4, Jan. 2016, doi: https://doi.org/10.58729/1941-6679.1296.

[7] S. C. Gupta and N. Goel, "Predictive modeling and analytics for diabetes using hyperparameter tuned machine learning techniques," *Procedia Computer Science*, vol. 218, pp. 1257–1269, 2023.

[8] M. Pal, "Random forest classifier for remote sensing classification," *International Journal of Remote Sensing*, vol. 26, no. 1, pp. 217–222, 2005.

[9] M. Sen *et al.*, "Evaluation of Water Quality Index Using Machine Learning Approach," *Lecture notes in electrical engineering*, pp. 401–408, Jan. 2023, doi: https://doi.org/10.1007/978-981-99-2710-4_33.

[10] M. F. Tanvir, M. M. Hossain, and M. A. Jishan, "Bayesian Regression for Predicting Subscription to Bank Term Deposits in Direct Marketing Campaigns," *arXiv preprint arXiv:2410.21539*, 2024.

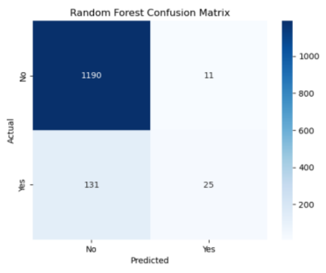
[11] J. Cervantes, F. Garcia-Lamont, L. Rodríguez-Mazahua, and A. Lopez, "A comprehensive survey on support vector machine classification: Applications, challenges and trends," *Neurocomputing*, vol. 408, no. 1, pp. 189–215, Sep. 2020, doi: https://doi.org/10.1016/j.neucom.2019.10.118.

[12] M. Pal, "Support vector machine-based feature selection for land cover classification: a case study with DAIS hyperspectral data," *International Journal of Remote Sensing*, vol. 27, no. 14, pp. 2877–2894, Jul. 2006, doi: https://doi.org/10.1080/01431160500242515.

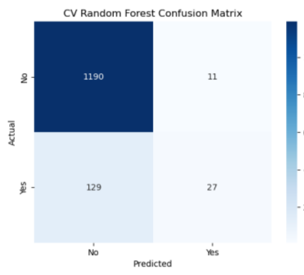
[13] M. A. Hearst, S. T. Dumais, E. Osuna, J. Platt, and B. Scholkopf, "Support vector machines," *IEEE Intelligent Systems and their Applications*, vol. 13, no. 4, pp. 18–28, Jul. 1998, doi: https://doi.org/10.1109/5254.708428.

[14] J. Dou *et al.*, "Improved landslide assessment using support vector machine with bagging, boosting, and stacking ensemble machine learning framework in a mountainous watershed, Japan," *Landslides*, vol. 17, no. 3, pp. 641–658, Oct. 2019, doi: https://doi.org/10.1007/s10346-019-01286-5.

[15] B. Pavlyshenko, "Using Stacking Approaches for Machine Learning Models," *IEEE Xplore*, Aug. 01, 2018. https://ieeexplore.ieee.org/abstract/document/8478522 (accessed Oct. 12, 2021).

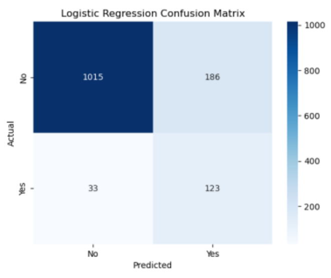


(i)

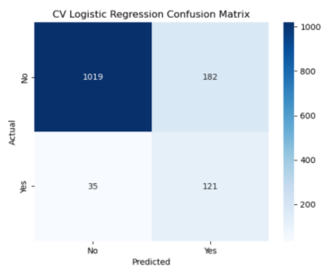


(ii)

(a) Random Forest

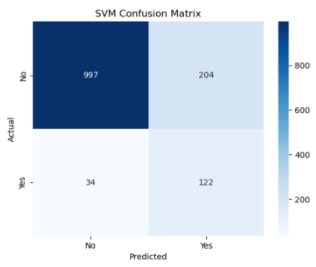


(i)

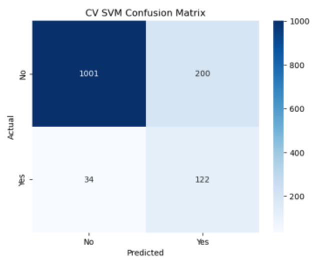


(ii)

(b) Logistic Regression

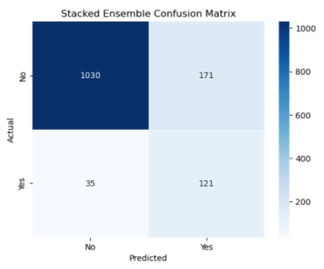


(i)

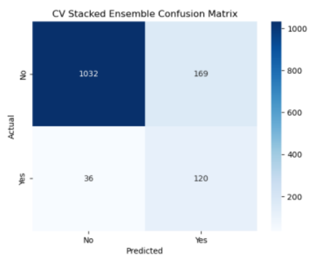


(ii)

(c) SVM Forest



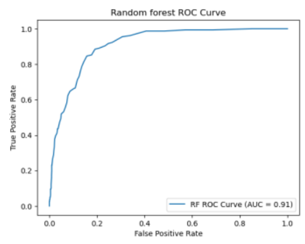
(i)



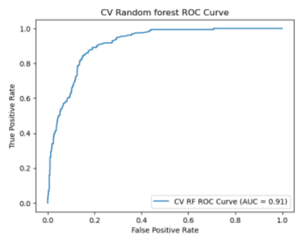
(ii)

(d) Stacked Ensemble

Fig 4. Models Confusion Matrices

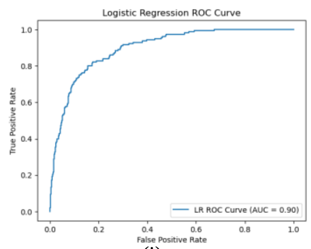


(i)

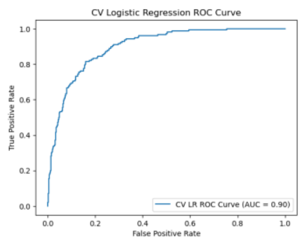


(ii)

(a) Random Forest

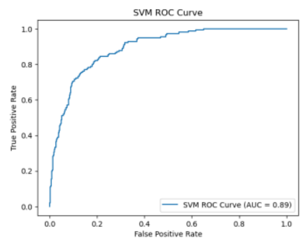


(i)

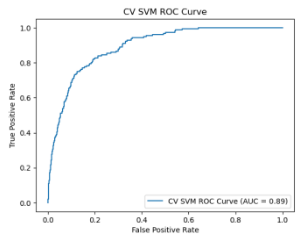


(ii)

(b) Logistic Regression

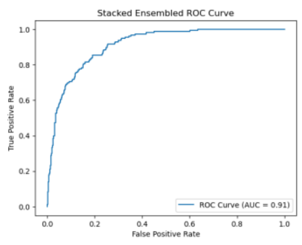


(i)

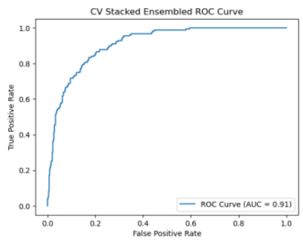


(ii)

(c) SVM Forest



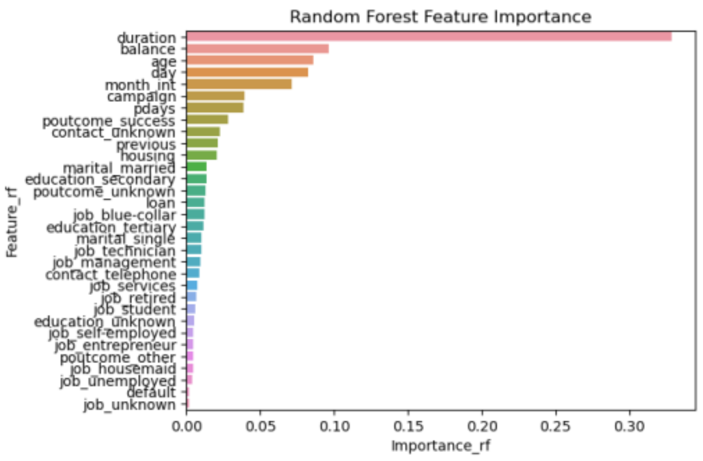
(i)



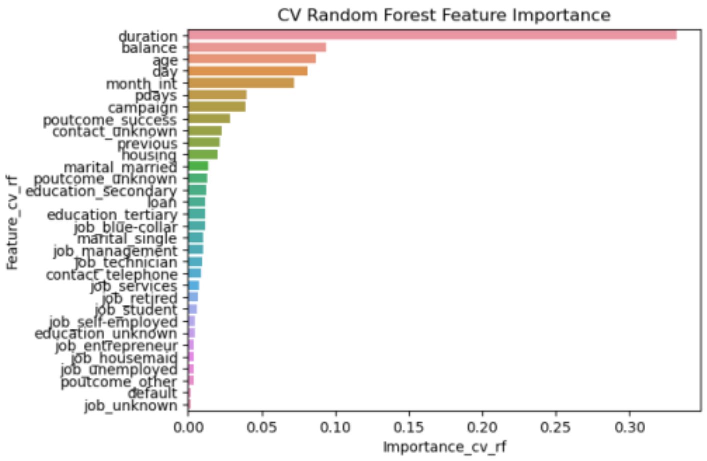
(ii)

(d) Stacked Ensemble

Fig 5. Models ROC Curves

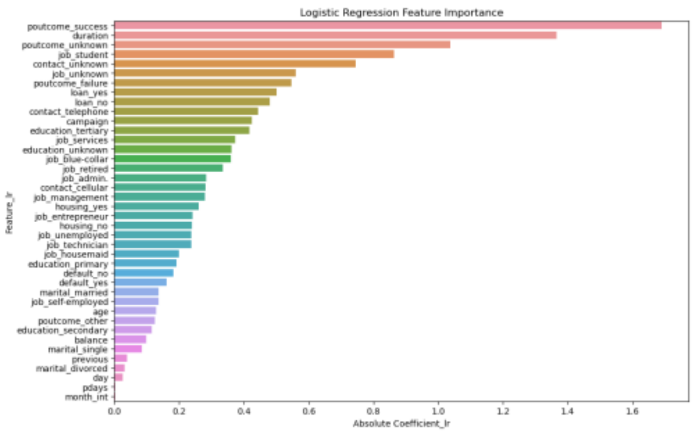


(i)

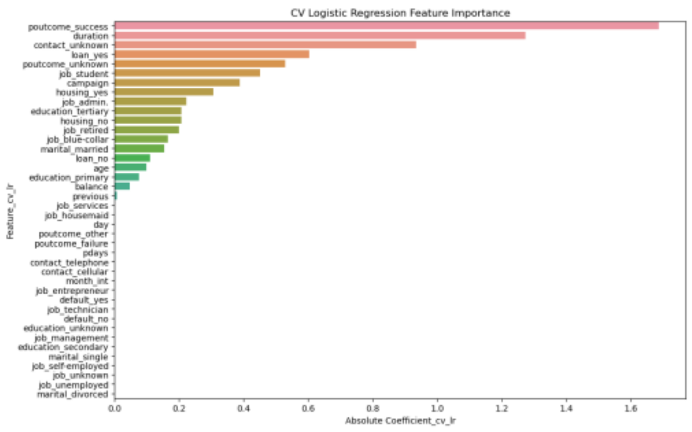


(ii)

(a) Random forest

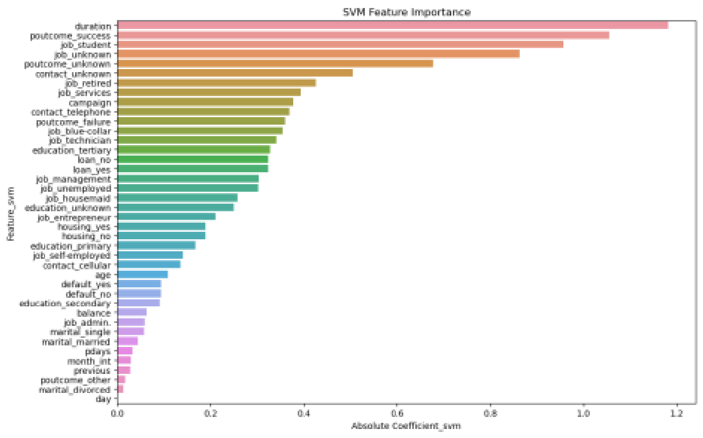


(i)

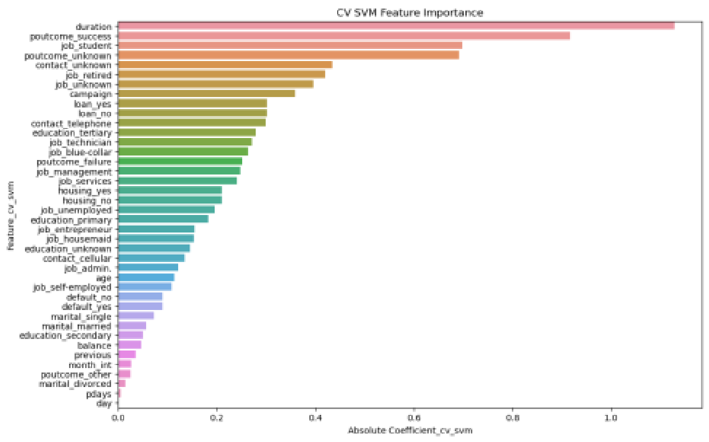


(ii)

(b) Logistic Regression



(i)



(ii)

(c) SVM

Fig. 6. Models' important features

APPENDIX A

TABLE II. DESCRIPTION OF THE DATASET FEATURES

Variable Name	Type	Description
age	Integer	Customers' age
job	Categorical	type of job (categorical: 'admin.','blue-collar','entrepreneur','housemaid','management','retired','self-employed','services','student','technician','unemployed','unknown')
marital	Categorical	marital status (categorical: 'divorced','married','single','unknown'; note: 'divorced' means divorced or widowed)
education	Categorical	(categorical: 'basic.4y','basic.6y','basic.9y','high.school','illiterate','professional.course','university.degree','unknown')
default	Binary	has credit in default?
balance	Integer	average yearly balance
housing	Binary	has housing loan?
loan	Binary	has personal loan?
contact	Categorical	contact communication type (categorical: 'cellular','telephone')
day_of_week	Date	last contact day of the week
month	Date	last contact month of year (categorical: 'jan', 'feb', 'mar', ..., 'nov', 'dec')
duration	Integer	last contact duration, in seconds (numeric). Important note: this attribute highly affects the output target (e.g., if duration=0 then y='no'). Yet, the duration is not known before a call is performed. Also, after the end of the call y is obviously known. Thus, this input should only be included for benchmark purposes and should be discarded if the intention is to have a realistic predictive model.
campaign	Integer	number of contacts performed during this campaign and for this client (numeric, includes last contact)
pdays	Integer	number of days that passed by after the client was last contacted from a previous campaign (numeric; -1 means client was not previously contacted)
previous	Integer	number of contacts performed before this campaign and for this client
poutcome	Categorical	outcome of the previous marketing campaign (categorical: 'failure','nonexistent','success')
y	Binary	has the client subscribed a term deposit?

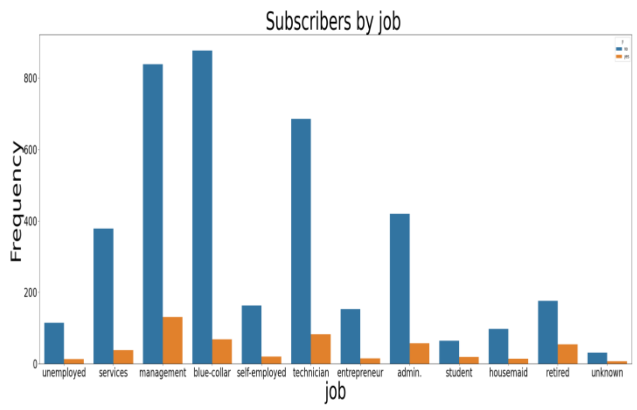
Dataset Link:

\bank+marketing\bank\bank.csv from

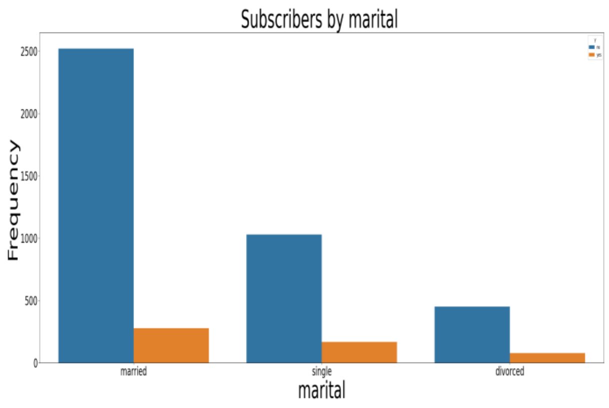
"https://archive.ics.uci.edu/static/public/222/bank+marketing.zip

APPENDIX B

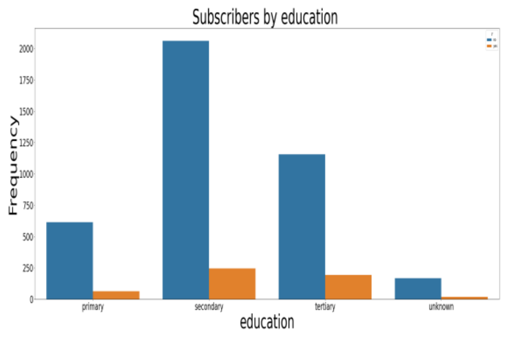
COUNT OF CUSTOMERS BY FEATURE PROPERTIES



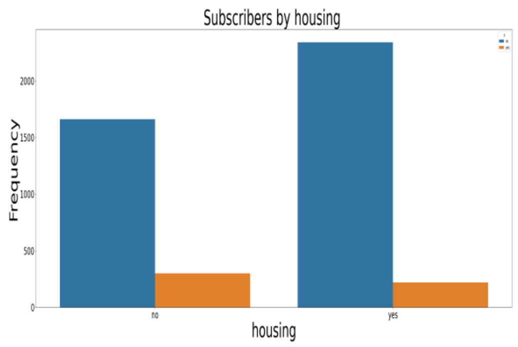
Value count of subscribers based on the Job property



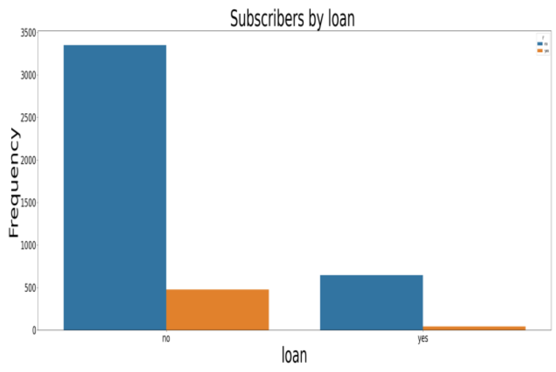
Value count of subscribers based on the Married property



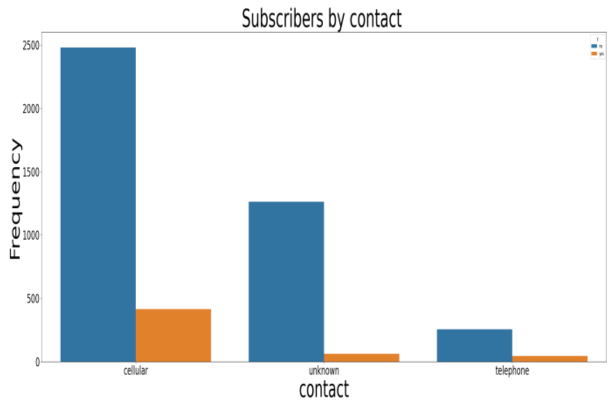
Value count of subscribers based on the Education property



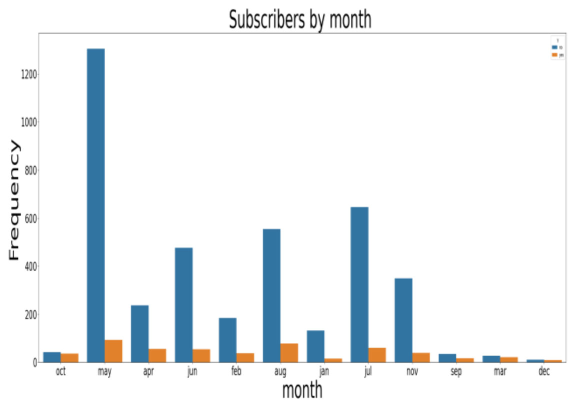
Value count of subscribers based on Housing status property



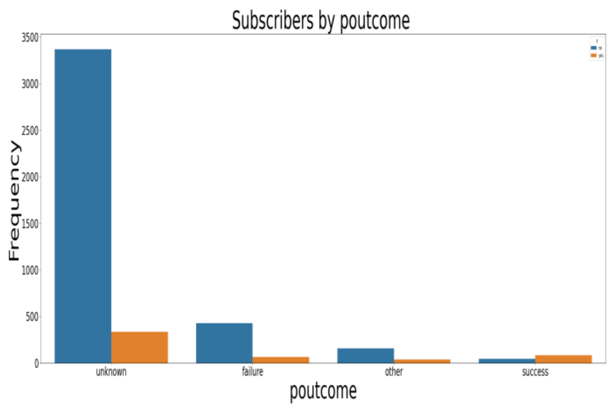
Value count of subscribers based on Loan status property



Value count of subscribers based on the Type of Contact property



Value count of subscribers based on the Month of contact property



Value count of subscribers based on the previous outcome property

APPENDIX C

SOURCE CODE

Comparative Study of Machine Learning Models for Predicting Term Deposit Subscriptions
The bank dataset is a collection of customer data used to predict term deposit subscriptions in marketing campaigns, featuring demographics, financial details, and campaign-related attributes. Random Forest, SVM, Logistic Regression, and an ensemble of Random Forest and Logistic Regression were applied on the dataset. By employing hyperparameter tuning, class imbalance handling, and feature importance analysis, the models performancec were evaluated to balance predictive accuracy, interpretability, and suitability for financial applications, offering insights into customer behavior and improved decision-making strategies.

```
# Importing the required libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.metrics import (accuracy_score, classification_report,
                             confusion_matrix, roc_auc_score, roc_curve,
                             average_precision_score, precision_recall_curve)
from math import log
from sklearn.preprocessing import StandardScaler, OneHotEncoder
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
from sklearn.ensemble import RandomForestClassifier, StackingClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
**EXPLORING THE DATASET

#Loading the bank dataset
data = pd.read_csv('bank.csv', delimiter=';')
data.head()

#A look into the dataset
data.info()

#Splitting the columns dataset into numerical and categorical columns bar the target column
numerical_features = ['age', 'balance', 'day', 'duration', 'campaign', 'pdays', 'previous']
categorical_features = ['job', 'marital', 'education', 'default', 'housing', 'loan', 'contact', 'month', 
'poutcome']

#Variation in the target variable
data['y'].value_counts(normalize=True)

#Visualising the target variable variation
sns.countplot(x='y', data=data)
plt.title('Variation in target variable classes')
plt.ylabel('Number of customers')
plt.xlabel('Subscribed')
plt.show()

#Visualising the relationship between the numerical features
sns.pairplot(data, hue='y')
plt.show()

#Visualising the dataset to see customers who subscribe by categories
plt.figure(figsize=(30, 100))
for i, col in enumerate(categorical_features):
    plt.subplot(len(categorical_features), 1, i+1)
    sns.countplot(data=data, x=col, hue='y')
    plt.title(f'Subscribers by {col} ', fontsize=50)
    plt.xlabel(col, fontsize=48)
    plt.ylabel('Frequency', fontsize=48)
    plt.xticks(fontsize=24)
    plt.yticks(fontsize=24)

plt.tight_layout()
plt.show()

#Generating a correlation heatmap for numerical columns
correlation_matrix = data.select_dtypes(include=[np.number]).corr()
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt=".2f", linewidths=0.5)
plt.title("Correlation Matrix Heatmap")
plt.show()
```

```
**GENERAL DATA PREPROCESSING

#Converting months to integer representation
month_map = {'jan':1, 'feb':2, 'mar':3, 'apr': 4, 'may': 5, 'jun':6, 'jul': 7, 'aug':8, 'sep': 9, 'oct':
10, 'nov': 11, 'dec': 12}

data['month_int'] = pd.Categorical(data['month'], categories=month_map.keys(), ordered=True).codes + 1
data['month_int'].value_counts()

#Dropping the month column
data.drop('month', axis=1, inplace=True)
data.info()

RANDOM FOREST MODEL
**PREPROCESSING FOR RANDOM FOREST
#making a copy of the dataset for random_forest
rf_data = data.copy()
rf_data.head()

#One-hot encoding for columns with labels
rf_data_encoded = pd.get_dummies(rf_data, columns=['job', 'marital', 'education', 'contact', 'poutcome'],
drop_first=True)
rf_data_encoded.head()

#Mapping binary columns into 0s and 1s
binary_columns = ['default', 'housing', 'loan', 'y']
rf_data_encoded[binary_columns] = rf_data_encoded[binary_columns].apply(lambda x: x.map({'yes': 1, 'no':
0}))
rf_data_encoded.head()

#Separating the rf_datset into features (X) and target (y)
X_rf = rf_data_encoded.drop('y', axis=1)
y_rf = rf_data_encoded['y']

#Splitting the rf_dataset into training and test data
X_train_rf, X_test_rf, y_train_rf, y_test_rf = train_test_split(X_rf,y_rf, test_size=0.3,
random_state=20, stratify=y_rf)

#Defining the model
rf_model = RandomForestClassifier(random_state=20, class_weight='balanced')
rf_model.fit(X_train_rf, y_train_rf)

#Random forest prediction
y_predict_rf = rf_model.predict(X_test_rf)
print(classification_report(y_test_rf, y_predict_rf))

#Confusion matrix for the random forest
rf_conf_matrix = confusion_matrix(y_test_rf, y_predict_rf)
sns.heatmap(rf_conf_matrix, annot=True, fmt='d', cmap='Blues', xticklabels=['No', 'Yes'],
yticklabels=['No', 'Yes'])
plt.title('Random Forest Confusion Matrix')
plt.ylabel('Actual')
plt.xlabel('Predicted')
plt.show()

#The probabilistic pridiction
y_rf_prob = rf_model.predict_proba(X_test_rf)[:, 1]

#ROC Curve for the random forest
roc_auc_rf = roc_auc_score(y_test_rf, y_rf_prob)
print(f"RF ROC-AUC Score: {roc_auc_rf}")

# Plot ROC Curve
fpr, tpr, _ = roc_curve(y_test_rf, y_rf_prob)
plt.plot(fpr, tpr, label=f"RF ROC Curve (AUC = {roc_auc_rf:.2f})")
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Random forest ROC Curve')
plt.legend()
plt.show()

#Extracting important features
importances_rf = rf_model.feature_importances_
feature_names_rf = X_rf.columns
importance_rf_df = pd.DataFrame({'Feature_rf': feature_names_rf, 'Importance_rf': importances_rf})
importance_rf_df = importance_rf_df.sort_values(by='Importance_rf', ascending=False)
importance_rf_df
```

```
#Visualising the important features
sns.barplot(x='Importance_rf', y='Feature_rf', data=importance_rf_df)
plt.title('Random Forest Feature Importance')
plt.show()

**Applying Hyperparameter tunning to Random Forest
#Defining the params
param_grid_rf = {
    'n_estimators': [100, 200, 300],
    'max_depth': [10, 20, 30],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4]
    }

#The model
grid_search_rf = GridSearchCV(estimator=rf_model, param_grid=param_grid_rf, cv=3, scoring='accuracy',
verbose=2, n_jobs=1)
grid_search_rf.fit(X_train_rf, y_train_rf)
print(f"Best Parameters: {grid_search_rf.best_params_}")
print(f"Best Score: {grid_search_rf.best_score_}")

#Predicting with the hyperparameter model
y_rftune_predict = grid_search_rf.predict(X_test_rf)
print(classification_report(y_test_rf, y_rftune_predict))

#Confusion matrix for the random forest
rftuned_conf_matrix = confusion_matrix(y_test_rf, y_rftune_predict)
sns.heatmap(rftuned_conf_matrix, annot=True, fmt='d', cmap='Blues', xticklabels=['No', 'Yes'],
yticklabels=['No', 'Yes'])
plt.title('CV Random Forest Confusion Matrix')
plt.ylabel('Actual')
plt.xlabel('Predicted')
plt.show()

#The tuned probabilistic pridiction
y_rftuned_prob = grid_search_rf.predict_proba(X_test_rf)[:, 1]

#ROC Curve for the hyper tunned random forest
roc_auc_rftuned = roc_auc_score(y_test_rf, y_rftuned_prob)
print(f"CV RF ROC-AUC Score: {roc_auc_rftuned}")

# Plot ROC Curve
fpr_rftune, tpr_rftune, _ = roc_curve(y_test_rf, y_rftuned_prob)
plt.plot(fpr_rftune, tpr_rftune, label=f"CV RF ROC Curve (AUC = {roc_auc_rftuned:.2f})")
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('CV Random forest ROC Curve')
plt.legend()
plt.show()

#Extracting CV RF important features
#Get the best estimator
best_estimator_rf = grid_search_rf.best_estimator_

#Access the feature importances
importances_cv_rf = best_estimator_rf.feature_importances_
feature_names_cv_rf = X_rf.columns
importance_cv_rf_df = pd.DataFrame({'Feature_cv_rf': feature_names_cv_rf, 'Importance_cv_rf':
importances_cv_rf})
importance_cv_rf_df = importance_cv_rf_df.sort_values(by='Importance_cv_rf', ascending=False)
importance_cv_rf_df

#Visualising the important features
sns.barplot(x='Importance_cv_rf', y='Feature_cv_rf', data=importance_cv_rf_df)
plt.title('CV Random Forest Feature Importance')
plt.show()

LOGISTIC REGRESSION MODEL
**PREPROCESSING FOR LOGISTIC REGRESSION
#Creating a copy of the dataset for logistic regression
lr_data = data.copy()
lr_data.head()

#Separating the lr dataset into feature (X) and target (y)
#Feature columns
X_lr = lr_data.drop(columns='y')

#Extracting target column and converting Yes and No labels to binary 1s and 0s
y_lr = lr_data['y'].apply(lambda x: 1 if x == 'yes' else 0)
```

```
lr_numerical_features = ['age', 'balance', 'day', 'duration', 'campaign', 'pdays', 'previous',
'month_int']
lr_categorical_features = ['job', 'marital', 'education', 'default', 'housing', 'loan', 'contact',
'poutcome']

#Scalling numerical features and encoding categorical features
lr_numerical_transformer = Pipeline(steps=[('scaler', StandardScaler())])

lr_categorical_transformer = Pipeline(steps=[('onehot', OneHotEncoder(handle_unknown='ignore'))])

lr_preprocessor = ColumnTransformer(transformers=[('num', lr_numerical_transformer,
lr_numerical_features),
                                               ('cat', lr_categorical_transformer,
lr_categorical_features)
                                               ])
#Splitting the lr dataset
X_lr_train, X_lr_test, y_lr_train, y_lr_test = train_test_split(X_lr, y_lr, test_size=0.3,
random_state=20, stratify=y_lr)

#defining the model
logistic_model = LogisticRegression(class_weight= 'balanced', solver='liblinear', random_state=20)

#Using a pipeline
lr_pipeline = Pipeline(steps=[('preprocessor', lr_preprocessor), ('classifier', logistic_model)])

#fitting the model
lr_pipeline.fit(X_lr_train, y_lr_train)

#predicting with the lr pipeline model
y_lr_predict = lr_pipeline.predict(X_lr_test)

#Probabilities for ROC-AUC
y_lr_prob = lr_pipeline.predict_proba(X_lr_test)[:, 1]

print(classification_report(y_lr_test, y_lr_predict))

#lr confusion matrix
lr_conf_matrix = confusion_matrix(y_lr_test, y_lr_predict)
sns.heatmap(lr_conf_matrix, annot=True, fmt='d', cmap='Blues', xticklabels=['No', 'Yes'],
yticklabels=['No', 'Yes'])
plt.title('Logistic Regression Confusion Matrix')
plt.ylabel('Actual')
plt.xlabel('Predicted')
plt.show()
lr_roc_auc = roc_auc_score(y_lr_test, y_lr_prob)
print(f"LR ROC-AUC Score: {lr_roc_auc}")

# Plot ROC Curve
fpr_lr, tpr_lr, _ = roc_curve(y_lr_test, y_lr_prob)
plt.plot(fpr_lr, tpr_lr, label=f"LR ROC Curve (AUC = {lr_roc_auc:.2f})")
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Logistic Regression ROC Curve')
plt.legend()
plt.show()

# Extract feature names after preprocessing
# Get feature names from the numerical and categorical transformers
lr_categorical_feature_new = lr_pipeline.named_steps['preprocessor'].\
    named_transformers_['cat'].get_feature_names_out(lr_categorical_features)

# Combine all feature names
all_lr_feature_names = np.concatenate([lr_numerical_features, lr_categorical_feature_new])

#Extract coefficeints
lr_coefficients = lr_pipeline.named_steps['classifier'].coef_[0]


lr_feature_importance = pd.DataFrame({
    'Feature_lr': all_lr_feature_names,
    'Coefficient_lr': lr_coefficients,
    'Absolute Coefficient_lr': np.abs(lr_coefficients)
    })

lr_feature_importance = lr_feature_importance.sort_values(by='Absolute Coefficient_lr', ascending=False)
lr_feature_importance

#Visualising the important features
plt.figure(figsize=(12, 8))
```

```
sns.barplot(x='Absolute Coefficient_lr', y='Feature_lr', data=lr_feature_importance)
plt.title('Logistic Regression Feature Importance')
plt.show()

**Applying Hyperparameter tunning to the Logistic Regression

#defining the params
lr_param_grid = {
    'classifier__C': [0.1, 1, 10, 100],
    'classifier__penalty': ['l1', 'l2']
    }

#Performing the Grid Search CV
lr_grid_search = GridSearchCV(lr_pipeline, param_grid=lr_param_grid, cv=5, scoring='accuracy', verbose=2,
n_jobs=-1)
lr_grid_search.fit(X_lr_train, y_lr_train)

#Best parameters and score
print(f"Best Parameters: {lr_grid_search.best_params_}")
print(f"Best Score: {lr_grid_search.best_score_}")
lr_best_model = lr_grid_search.best_estimator_
y_lr_best_predict = lr_best_model.predict(X_lr_test)
y_lr_best_prob = lr_best_model.predict_proba(X_lr_test)[:, 1]

print(classification_report(y_lr_test, y_lr_best_predict))

#lr tunned confusion matrix
lrtune_cm = confusion_matrix(y_lr_test, y_lr_best_predict)
sns.heatmap(lrtune_cm, annot=True, fmt='d', cmap='Blues', xticklabels=['No', 'Yes'], yticklabels=['No',
'Yes'])
plt.title('CV Logistic Regression Confusion Matrix')
plt.ylabel('Actual')
plt.xlabel('Predicted')
plt.show()
lr_cv_roc_auc = roc_auc_score(y_lr_test, y_lr_best_prob)
print(f"CV LR ROC-AUC Score: {lr_cv_roc_auc}")

# Plot ROC Curve
lr_cv_fpr, lr_cv_tpr, _ = roc_curve(y_lr_test, y_lr_best_prob)
plt.plot(lr_cv_fpr, lr_cv_tpr, label=f"CV LR ROC Curve (AUC = {lr_cv_roc_auc:.2f})")
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('CV Logistic Regression ROC Curve')
plt.legend()
plt.show()

#Extract coefficeints
cv_lr_coefficients = lr_best_model.named_steps['classifier'].coef_[0]


cv_lr_feature_importance = pd.DataFrame({
    'Feature_cv_lr': all_lr_feature_names,
    'Coefficient_cv_lr': cv_lr_coefficients,
    'Absolute Coefficient_cv_lr': np.abs(cv_lr_coefficients)
    })

cv_lr_feature_importance = cv_lr_feature_importance.sort_values(by='Absolute Coefficient_cv_lr',
ascending=False)
cv_lr_feature_importance

#Visualising the important features
plt.figure(figsize=(12, 8))
sns.barplot(x='Absolute Coefficient_cv_lr', y='Feature_cv_lr', data=cv_lr_feature_importance)
plt.title('CV Logistic Regression Feature Importance')
plt.show()
SVM MODEL

#Creating a copy of the dataset for SVM
svm_data = data.copy()
svm_data.head()

#Separating the SVM dataset into feature (X) and target (y)
#Feature columns
X_svm = svm_data.drop(columns='y')

#Extracting target column and converting Yes and No labels to binary 1s and 0s
y_svm = svm_data['y'].apply(lambda x: 1 if x == 'yes' else 0)
svm_numerical_features = ['age', 'balance', 'day', 'duration', 'campaign', 'pdays', 'previous',
'month_int']
```

```
svm_categorical_features = ['job', 'marital', 'education', 'default', 'housing', 'loan', 'contact',
'poutcome']

#Scalling numerical features and encoding categorical features
svm_numerical_transformer = Pipeline(steps=[('scaler', StandardScaler())])

svm_categorical_transformer = Pipeline(steps=[('onehot', OneHotEncoder(handle_unknown='ignore'))])

svm_preprocessor = ColumnTransformer(transformers=[('num', svm_numerical_transformer,
svm_numerical_features),
                                               ('cat', svm_categorical_transformer,
svm_categorical_features)
                                               ])
#Splitting the lr dataset
X_svm_train, X_svm_test, y_svm_train, y_svm_test = train_test_split(X_svm, y_svm, test_size=0.3,
random_state=20, stratify=y_lr)

#defining the model
svm_model = SVC(kernel='linear', class_weight='balanced', probability=True, random_state=20)

# Create an SVM pipeline
svm_pipeline = Pipeline(steps=[
    ('preprocessor', svm_preprocessor),
    ('svm', svm_model)
    ])

# Train with the model
svm_pipeline.fit(X_svm_train, y_svm_train)

#predicting with the svm pipeline model

y_svm_predict = svm_pipeline.predict(X_svm_test)

#Probabilities for ROC-AUC
y_svm_prob = svm_pipeline.predict_proba(X_svm_test)[:, 1]

print(classification_report(y_svm_test, y_svm_predict))

#svm confusion matrix
svm_conf_matrix = confusion_matrix(y_svm_test, y_svm_predict)
sns.heatmap(svm_conf_matrix, annot=True, fmt='d', cmap='Blues', xticklabels=['No', 'Yes'],
yticklabels=['No', 'Yes'])
plt.title('SVM Confusion Matrix')
plt.ylabel('Actual')
plt.xlabel('Predicted')
plt.show()
svm_roc_auc = roc_auc_score(y_svm_test, y_svm_prob)
print(f"SVM ROC-AUC Score: {svm_roc_auc}")

# Plot ROC Curve
fpr_svm, tpr_svm, _ = roc_curve(y_svm_test, y_svm_prob)
plt.plot(fpr_svm, tpr_svm, label=f"SVM ROC Curve (AUC = {svm_roc_auc:.2f})")
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('SVM ROC Curve')
plt.legend()
plt.show()

# Extract feature names after preprocessing
# Get feature names from the numerical and categorical transformers
svm_categorical_feature_new = svm_pipeline.named_steps['preprocessor'].\
    named_transformers_['cat'].get_feature_names_out(svm_categorical_features)

# Combine all feature names
all_svm_feature_names = np.concatenate([svm_numerical_features, svm_categorical_feature_new])

#Extract coefficeints
svm_coefficients = svm_pipeline.named_steps['svm'].coef_[0]

svm_feature_importance = pd.DataFrame({
    'Feature_svm': all_svm_feature_names,
    'Coefficient_svm': svm_coefficients,
    'Absolute Coefficient_svm': np.abs(svm_coefficients)
    })

svm_feature_importance = svm_feature_importance.sort_values(by='Absolute Coefficient_svm',
ascending=False)
svm_feature_importance
```

```
#Visualising the important features
plt.figure(figsize=(12, 8))
sns.barplot(x='Absolute Coefficient_svm', y='Feature_svm', data=svm_feature_importance)
plt.title('SVM Feature Importance')
plt.show()

**Applying Hyperparameter tunning to the SVM model
# Define parameter grid for hyperparameter tuning
svm_param_grid = {
    'svm__C': [0.1, 1, 10, 100]
    }

# Performing the SVM Grid Search CV
svm_grid_search = GridSearchCV(svm_pipeline, svm_param_grid, cv=3, scoring='accuracy', verbose=2,
n_jobs=-1)
svm_grid_search.fit(X_svm_train, y_svm_train)

#Best parameters and score
print(f"SVM Best Parameters: {svm_grid_search.best_params_}")
print(f"SVM Best Score: {svm_grid_search.best_score_}")

# Predict and evaluate
svm_best_model = svm_grid_search.best_estimator_
y_svm_best_pred = svm_best_model.predict(X_svm_test)
y_svm_best_prob = svm_best_model.predict_proba(X_svm_test)[:, 1]

print(classification_report(y_svm_test, y_svm_best_pred))

#svm tunned confusion matrix
svm_cv_cm = confusion_matrix(y_svm_test, y_svm_best_pred)
sns.heatmap(svm_cv_cm, annot=True, fmt='d', cmap='Blues', xticklabels=['No', 'Yes'], yticklabels=['No',
'Yes'])
plt.title('CV SVM Confusion Matrix')
plt.ylabel('Actual')
plt.xlabel('Predicted')
plt.show()
svm_cv_roc_auc = roc_auc_score(y_svm_test, y_svm_best_prob)
print(f"CV SVM ROC-AUC Score: {svm_cv_roc_auc}")

# Plot ROC Curve
svm_cv_fpr, svm_cv_tpr, _ = roc_curve(y_svm_test, y_svm_best_prob)
plt.plot(svm_cv_fpr, svm_cv_tpr, label=f"CV SVM ROC Curve (AUC = {svm_cv_roc_auc:.2f})")
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('CV SVM ROC Curve')
plt.legend()
plt.show()

#Extract coefficeints

cv_svm_coefficients = svm_best_model.named_steps['svm'].coef_[0]


cv_svm_feature_importance = pd.DataFrame({
    'Feature_cv_svm': all_svm_feature_names,
    'Coefficient_cv_svm': cv_svm_coefficients,
    'Absolute Coefficient_cv_svm': np.abs(cv_svm_coefficients)
    })

cv_svm_feature_importance = cv_svm_feature_importance.sort_values(by='Absolute Coefficient_cv_svm',
ascending=False)
cv_svm_feature_importance

#Visualising the important features
plt.figure(figsize=(12, 8))
sns.barplot(x='Absolute Coefficient_cv_svm', y='Feature_cv_svm', data=cv_svm_feature_importance)
plt.title('CV SVM Feature Importance')
plt.show()

STACKED RANDOM FOREST AND LOGISTIC REGRESSION ENSEMBLE
#Creating a copy of the dataset for the ensemble model
en_data = data.copy()
en_data.head()

#Separating the lr dataset into feature (X) and target (y)
#Feature columns
X_en = en_data.drop(columns='y')
```

```
#Extracting target column and converting Yes and No labels to binary 1s and 0s
y_en = en_data['y'].apply(lambda x: 1 if x == 'yes' else 0)
en_numerical_features = ['age', 'balance', 'day', 'duration', 'campaign', 'pdays', 'previous',
'month_int']
en_categorical_features = ['job', 'marital', 'education', 'default', 'housing', 'loan', 'contact',
'poutcome']

#Scalling numerical features and encoding categorical features

en_numerical_transformer = Pipeline(steps=[('scaler', StandardScaler())])

en_categorical_transformer = Pipeline(steps=[('onehot', OneHotEncoder(handle_unknown='ignore'))])

en_preprocessor = ColumnTransformer(transformers=[('num_en', en_numerical_transformer,
en_numerical_features),
                                               ('cat_en', en_categorical_transformer,
en_categorical_features)
                                               ])
#Pipeline for the base models

random_forest_model = Pipeline(steps=[
    ('en_preprocessor', en_preprocessor),
     ('rf', RandomForestClassifier(n_estimators =100, random_state=20))
     ])

logistic_regression_model = Pipeline(steps=[
    ('en_preprocessor', en_preprocessor),
     ('lr', logistic_model)
    ])

#combining the models together with stacking ensemble
stacking_clf = StackingClassifier(
    estimators = [
    ('rf', random_forest_model),
    ('lr', logistic_regression_model)
  ],
    final_estimator = LogisticRegression(class_weight='balanced', solver='liblinear', random_state=20)
)

#Splitting dataset
X_train_en, X_test_en, y_train_en, y_test_en = train_test_split(X_en, y_en, test_size=0.3,
random_state=20, stratify=y_en)

#Fitting the stacked model
stacking_clf.fit(X_train_en, y_train_en)

#Prediction with the ensemble
y_stack_predict = stacking_clf.predict(X_test_en)

print(classification_report(y_test_en, y_stack_predict))

#Confusion matrix for the ensemble
stack_conf_matrix = confusion_matrix(y_test_en, y_stack_predict)
sns.heatmap(stack_conf_matrix, annot=True, fmt='d', cmap='Blues', xticklabels=['No', 'Yes'],
yticklabels=['No', 'Yes'])
plt.title('Stacked Ensemble Confusion Matrix')
plt.ylabel('Actual')
plt.xlabel('Predicted')
plt.show()
y_en_stack_prob = stacking_clf.predict_proba(X_test_en)[:, 1]

en_roc_auc = roc_auc_score(y_test_en, y_en_stack_prob)
print(f"Stacked_en_ROC-AUC Score: {en_roc_auc}")

# Plot ROC Curve
en_fpr, en_tpr, _ = roc_curve(y_test_en, y_en_stack_prob)
plt.plot(en_fpr, en_tpr, label=f"ROC Curve (AUC = {en_roc_auc:.2f})")
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Stacked Ensembled ROC Curve')
plt.legend()
plt.show()

**Applying Hyperparameter tunning to the Ensemble Model

#Splitting the dataset for ensemble model
X_train_ent, X_test_ent, y_train_ent, y_test_ent = train_test_split(X_en, y_en, test_size=0.3,
random_state=20, stratify=y_en)
#Defining the params
```

```
stack_param_grid = {

    #Random FOrest parameters in the base model pipeline
    'rf__rf__n_estimators': [100, 200, 300],
    'rf__rf__max_depth': [10, 20, 30],

    #Logistic Regression parameters in the base model pipeline
    'lr__lr__C': [0.1, 1, 10],

    #Final estimator parameters
    'final_estimator__C': [0.1, 1, 10]
  }

#Defining the tuned stacked model
stack_grid_search = GridSearchCV(estimator=stacking_clf, param_grid=stack_param_grid, cv=3,
scoring='accuracy', verbose=2, n_jobs=-1)


#Fit the Training data
stack_grid_search.fit(X_train_ent, y_train_ent)
print(f"Best Parameters: {stack_grid_search.best_params_}")
print(f"Best Score: {stack_grid_search.best_score_}")
stack_best_model = stack_grid_search.best_estimator_
stack_best_predict = stack_best_model.predict(X_test_ent)
stack_best_prob = stack_best_model.predict_proba(X_test_ent)[:, 1]


print(classification_report(y_test_ent, stack_best_predict))
stackgrid_conf_matrix = confusion_matrix(y_test_ent, stack_best_predict)
sns.heatmap(stackgrid_conf_matrix, annot=True, fmt='d', cmap='Blues', xticklabels=['No', 'Yes'],
yticklabels=['No', 'Yes'])
plt.title('CV Stacked Ensemble Confusion Matrix')
plt.ylabel('Actual')
plt.xlabel('Predicted')
plt.show()
ent_roc_auc = roc_auc_score(y_test_ent, stack_best_prob)
print(f"ROC-AUC Score: {ent_roc_auc}")

# Plot ROC Curve
ent_fpr, ent_tpr, _ = roc_curve(y_test_ent, stack_best_prob)
plt.plot(ent_fpr, ent_tpr, label=f"ROC Curve (AUC = {ent_roc_auc:.2f})")
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('CV Stacked Ensembled ROC Curve')
plt.legend()
plt.show()
```