Georgia State University

CSC4780/6780&DSCI4780 – Fundamentals of Data Science

[2022]

Final Project Report

[BANK DATA ANALYSIS]

[AJKFDS]
Aryan Shrestha
Jagan Gedela
Sridutt Kalidindi
Caleb Tsai
Christopher Yoon

Table of Contents

1. Business Understanding 3
1.1 Business Problem 3
1.2 Dataset 3
1.3 Proposed Analytics Solution 3
2. Data Exploration and Preprocessing 4
2.1 Data Quality Report 4
2.2 Missing Values and Outliers 11
2.3 Normalization 11
2.4 Transformations 12
2.5 Feature Selection 13
2.5.1 Filter Method 13
2.5.2 Wrapper Method 13
3. Model Selection and Evaluation 13
3.1 Evaluation Metrics 13
3.2 Models 13
3.3 Evaluation 14
3.3.1 Evaluation Settings and Sampling 14
3.3.2 Hyperparameter Optimization 14
3.3.3 Evaluation 14
3.3.3.1 Naive Bayes 14
3.3.3.2 Logistic Regression 15
3.3.3.3 ADABoost 15
3.3.3.4 Support Vector Machine (SVM) 16
3.3.3.5 Random Forest 16
4 Results and Conclusion 18

1. Business Understanding

This project assists banks in determining the factors that influence whether a consumer will accept a bank's term deposit offer. The banks can use this information to identify consumers who are likely to accept the term deposit and lower the effort required because they are likely to accept it. They can also provide extra incentives to less likely consumers to accept. The project's purpose is to find crucial attributes in the dataset that can help categorize clients based on whether or not they accept or reject the term deposit that has been provided to them.

1.1 Business Problem

Banks routinely contact many customers by phone or email to offer term plans. The majority of these calls, it is thought, do not result in the anticipated outcome. As a result, banks' time and resources are spent in vain. We present a model that can predict which users are more likely to accept term policies based on some basic user information to address this problem. The model may also determine the optimal marketing plan for the next campaign.

1.2 Dataset

Data is gathered through a phone call campaign for term deposits. The campaign aims to give clients a term deposit while recording other customer attributes. The information also includes the call's conclusion, such as whether the customer accepted or rejected the term deposit. Using this property, we can turn a real-world problem into a Supervised Machine Learning challenge.

The dataset has 21 attributes and 41,189 instances. This dataset has both continuous and categorical (ordinal and nominal) features, making it an excellent candidate for applying CRISP-Data Mining techniques. This dataset also allows several algorithms to address missing values and detect outliers. Using the cleaned dataset, various data modeling techniques can be used to identify the link between characteristics and goal variables.

1.3 Proposed Analytics Solution

For this dataset, we will run through various classification techniques such as Logistic Regression, Random Forest, Support Vector Machine, and KNN. Before classification, however, the data will be cleaned and analyzed through statistical analysis such as Exploratory Data Analysis, Principal Component Analysis, Factor Analysis, and Correlations. Relating to this is the subject of identifying missing or outlier values, which may skew data or disrupt the accuracy of our analysis, so to pre-process our data, we should get rid of these values and replace them accordingly with methods such as mean and mode replacement. After the initial cleaning of the data, normalization, and transformation occurs to ensure that the data is finally ready for our classification.

2. Data Exploration and Preprocessing

For data visualization techniques, exploratory data analysis (EDA) has been used to analyze and examine data sets and summarize their key features. It also helps determine if the statistical approaches considered for data analysis are appropriate.

2.1 Data Quality Report

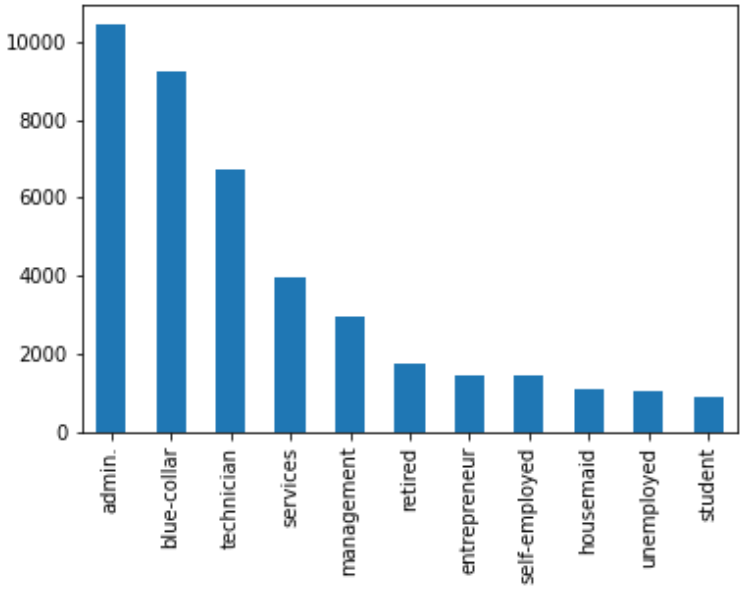
From the Data Quality Reports, we can describe the state of the features given by the dataset and determine whether they are fine as-is or if they should be split, transformed, and normalized. Generally, missing values on the data are inconsequential, as most features do not have more than 10% missing values. However, the specific feature pdays includes more than 96% of its values as missing values. In this case, we might drop it entirely or change the feature from continuous to categorical.

Table 1. Data Quality Report for Categorical Features

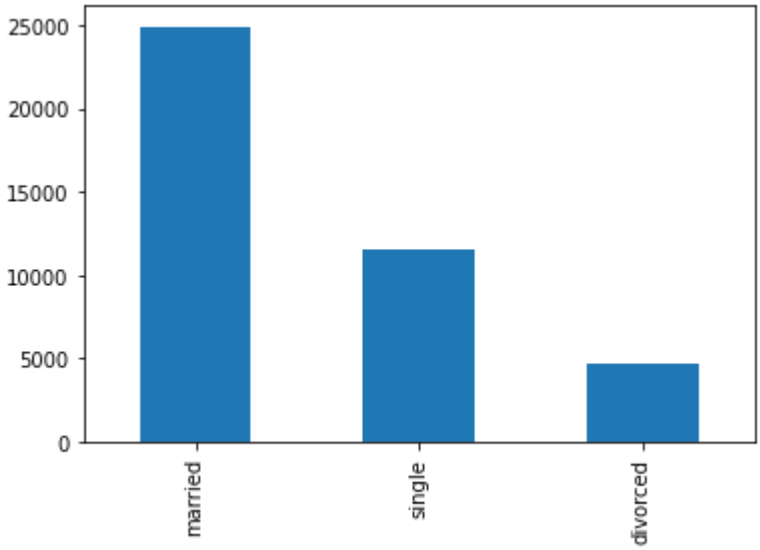
Feature	% of Missing	2nd Mode	2nd Mode Freq	2nd Mode Perc	Card	Count	Mode	Mode %	Mode Freq
job	0.80	blue-collar	9253.0	22.47	12.0	41176.0	admin.	25.30	10419.0
marital	0.19	single	11564.0	28.08	4.0	41176.0	married	60.52	24921.0
education	4.20	high.school	9512.0	23.10	8.0	41176.0	university.degree	29.54	12164.0
default	20.88	yes	3.0	0.01	3.0	41176.0	no	79.12	32577.0
housing	2.40	no	18615.0	45.21	3.0	41176.0	yes	52.39	21571.0
loan	2.40	yes	6248.0	15.17	3.0	41176.0	no	82.42	33938.0
contact	0.00	telephone	15041.0	36.53	2.0	41176.0	cellular	63.47	26135.0
month	0.00	jul	7169.0	17.41	10.0	41176.0	may	33.43	13767.0
day_of_week	0.00	mon	8512.0	20.67	5.0	41176.0	thu	20.93	8618.0
poutcome	0.00	failure	4252.0	10.33	3.0	41176.0	nonexistent	86.34	35551.0
y	0.00	yes	4639.0	11.27	2.0	41176.0	no	88.73	36537.0

Figure 1. Visualizations of Categorical Features in Dataset

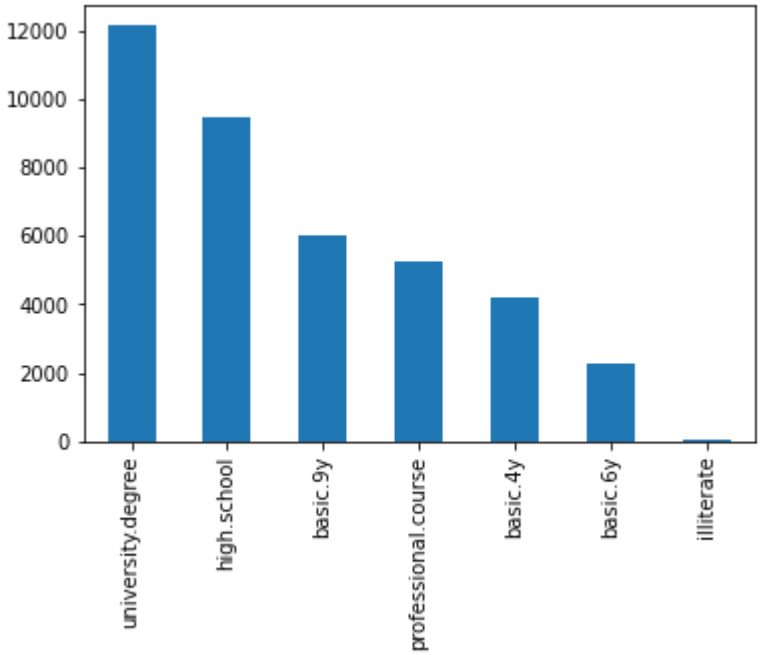
job



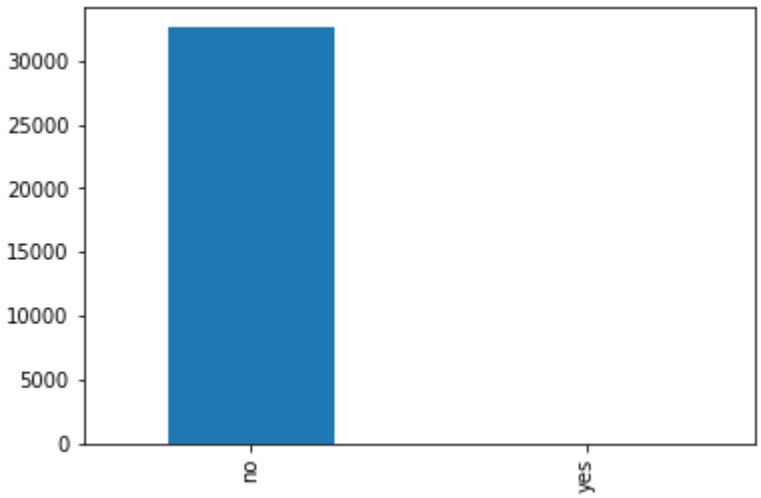
marital



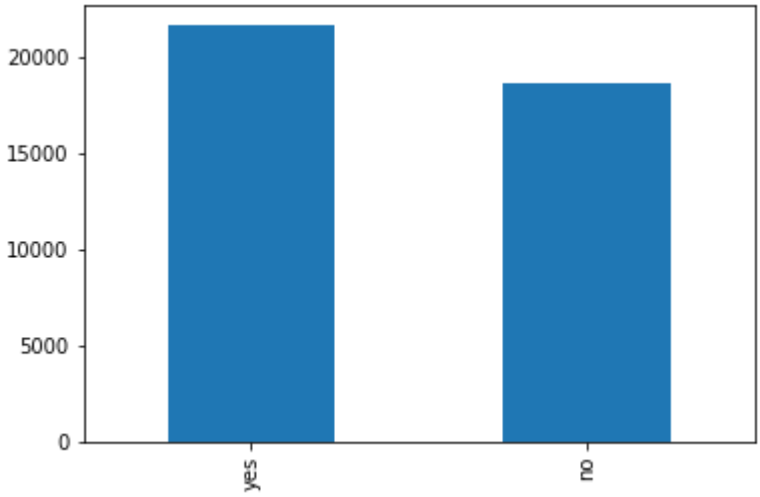
education



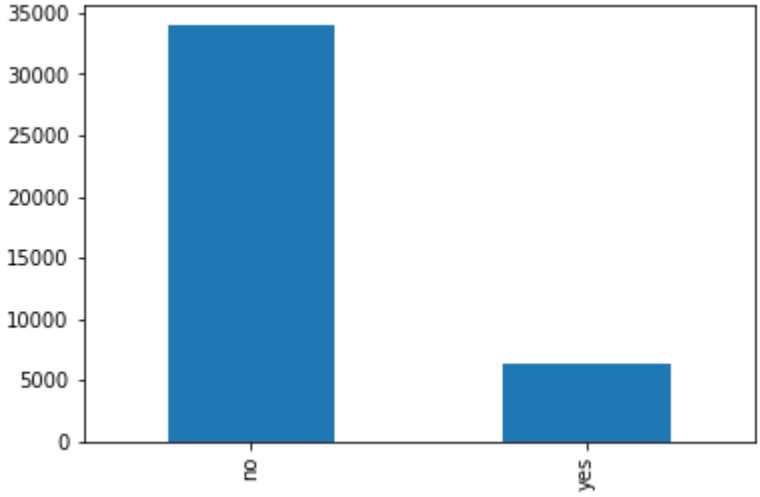
default



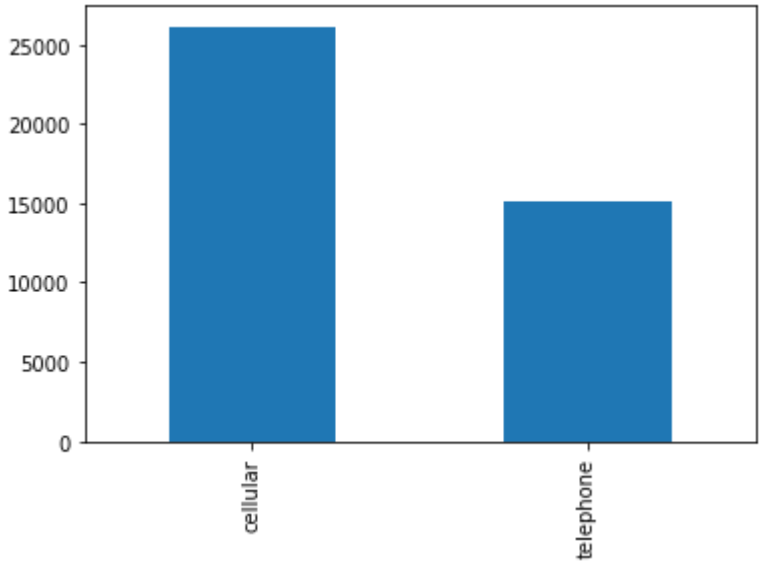
housing



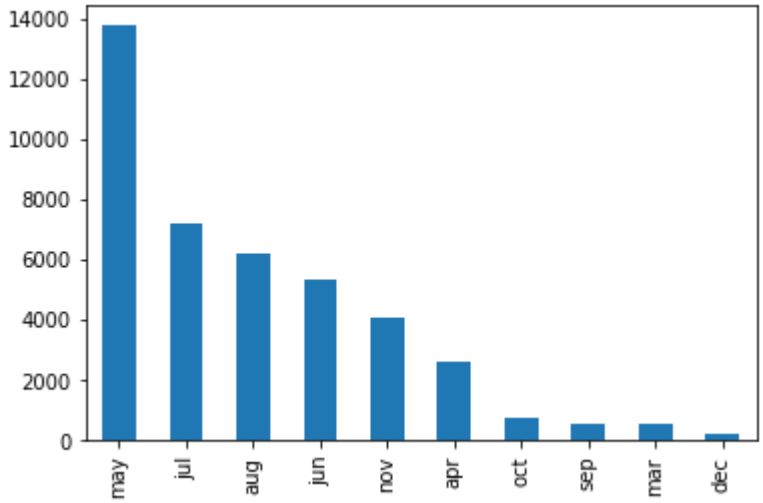
loan



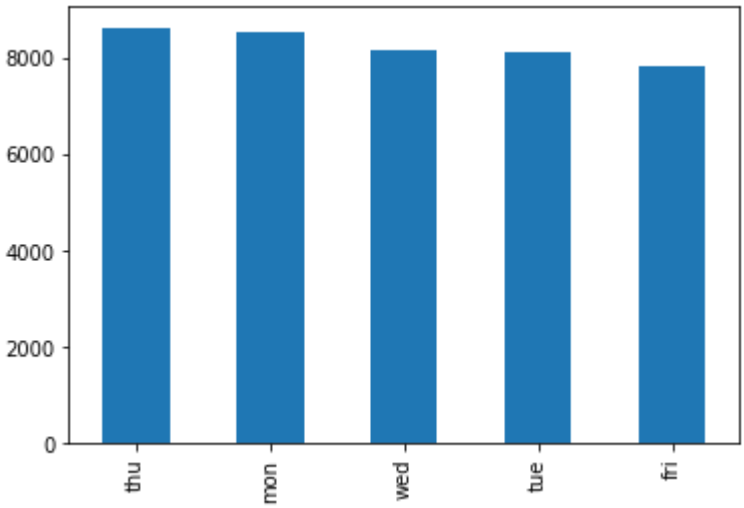
contact



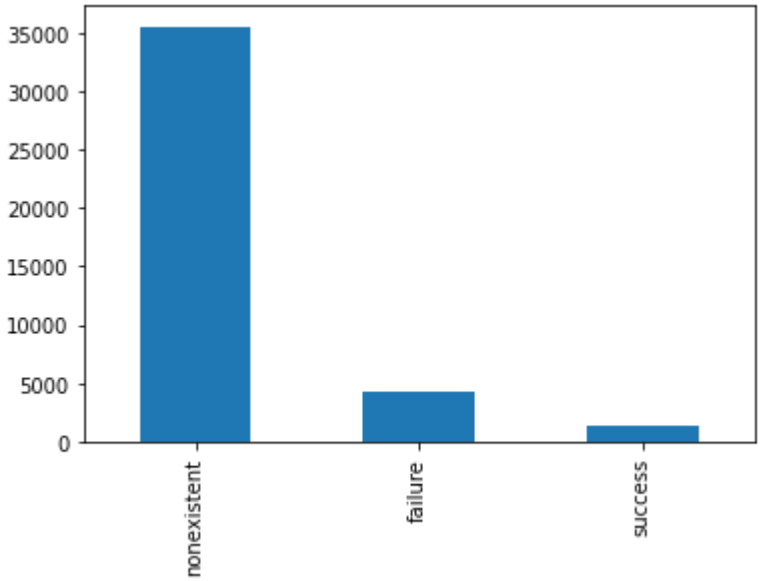
month



day_of_week



poutcome



y

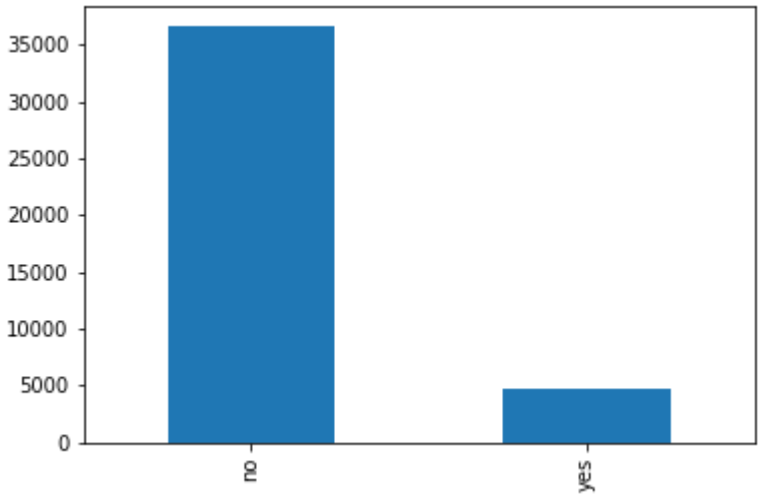
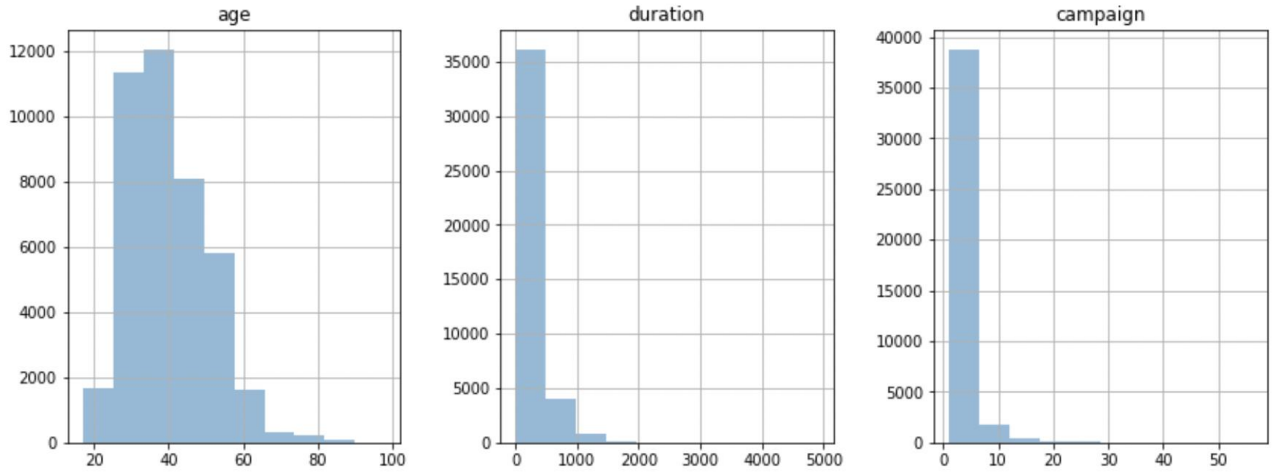
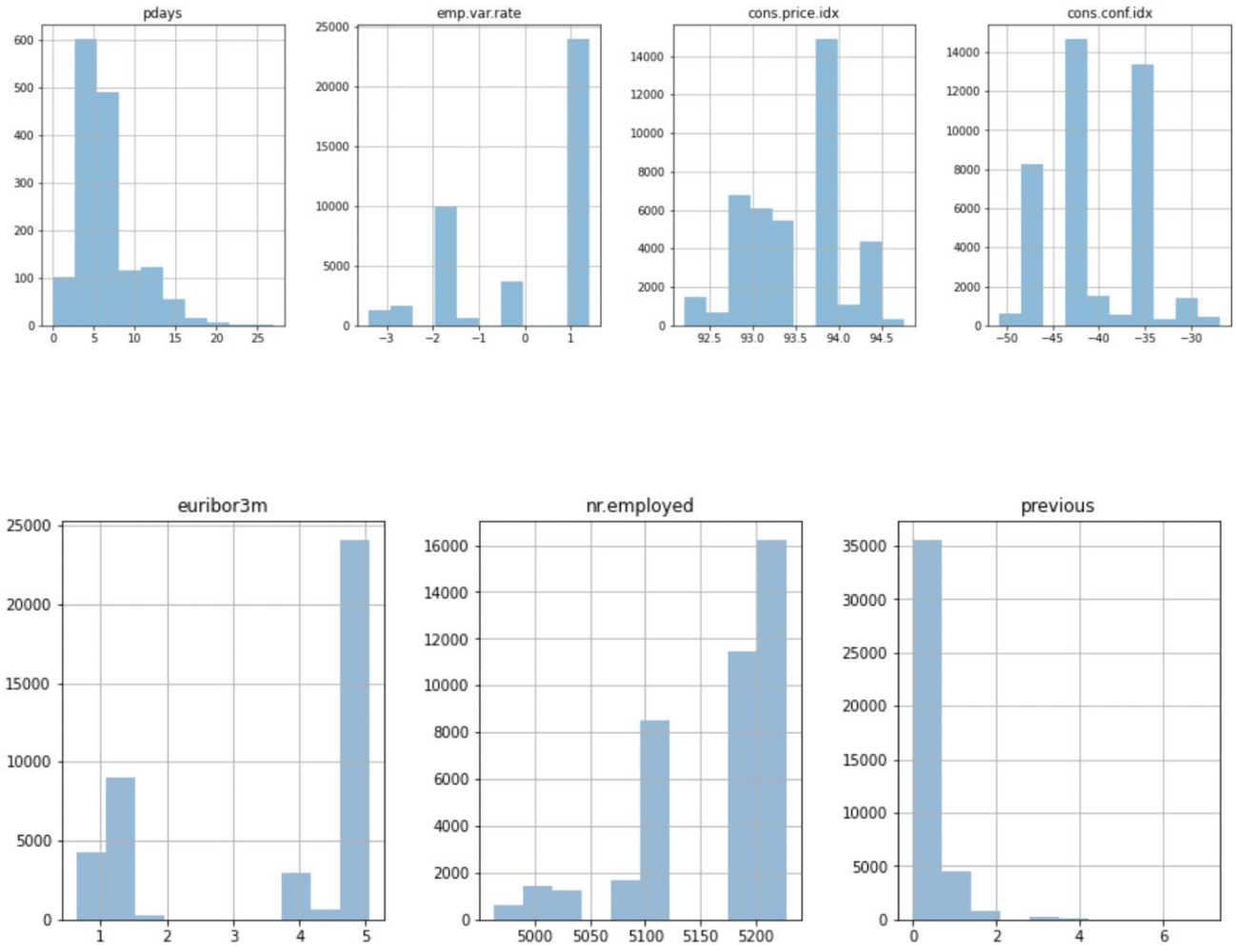


Table 2. Data Quality Report for Continuous Features

Feature	% of Missing	Card.	Count	Max	Mean	Median	Min.	Q1	Q3	Std. Dev.
age	0.00	78.0	41176.0	98.000	40.023800	38.000	17.000	32.000	47.000	10.420680
duration	0.00	1544.0	41176.0	4918.000	258.279837	180.000	0.000	102.000	319.000	259.260227
campaign	0.00	42.0	41176.0	56.000	2.567879	2.000	1.000	1.000	3.000	2.770318
pdays	96.32	27.0	41176.0	27.000	6.014521	6.000	0.000	3.000	7.000	3.824906
emp.var.rate	0.00	10.0	41176.0	1.400	0.081922	1.100	-3.400	-1.800	1.400	1.570883
cons.price.idx	0.00	26.0	41176.0	94.767	93.575720	93.749	92.201	93.075	93.994	0.578839
cons.conf.idx	0.00	26.0	41176.0	-26.900	-40.502863	-41.800	-50.800	-42.700	-36.400	4.627860
euribor3m	0.00	316.0	41176.0	5.045	3.621293	4.857	0.634	1.344	4.961	1.734437
nr.employed	0.00	11.0	41176.0	5228.100	5167.034870	5191.000	4963.600	5099.100	5228.100	72.251364
previous	0.00	8.0	41176.0	7.000	0.173013	0.000	0.000	0.000	0.000	0.494964

Figure 1. Visualizations of Continuous Features in Dataset





2.2 Missing Values and Outliers

To increase the performance of our classifiers, we need to replace or remove all missing values and identify outliers to see if they should be replaced/removed as well. Only pdays have missing values for continuous features, perhaps because there are no calls on record between the customer and the bank. For these missing values, we replace the NaN values with 0 for now. However, a different solution may be used to improve this feature if the accuracy of this replacement affects classification. For categorical features, quite a few have missing values- though since a couple has only less than 1 percent missing values, the values are dropped using dropna instead of simply filling the values. For those categorical features with more than 1 percent missing values, the missing values are replaced by the mode to prevent conflicts with our classifier's outcomes.

2.3 Normalization

During the data normalization, the values for continuous data in the dataset are scaled between 0 and 1 based on each column's minimum and maximum values. During normalization, it can be more apparent

to the viewer that specific columns have values that generally reach either 0 or 1 after the process, which makes it more straightforward when calculating the weight of each particular value. Initial feature ranges such as the range for the duration [0, 4918] and even age [17, 98] are the features that benefit the most from this process, as the data that can be visualized by normalizing them is more concise and easier to read on graphs. Most normalized/continuous features in this dataset are usually spread out across only a few specific values. However, they may benefit from the normalization less- but the process will create more concise and readable data regardless.

```
In [24]: scaler = preprocessing.MinMaxScaler()

         d = scaler.fit_transform(transformed_data)

         scaled_df = pd.DataFrame(d, columns=transformed_data.columns)


         scaled_df
```

Out[24]:

	age	duration	campaign	pdays	previous	emp.var.rate	cons.price.idx	cons.conf.idx	euribor3m	nr.employed	...	day_of_week_fri	day_of_week_mo
0	0.481481	0.053070	0.000000	0.0	0.000000	0.937500	0.698753	0.60251	0.957379	0.859735	...	0.0	1
1	0.493827	0.030297	0.000000	0.0	0.000000	0.937500	0.698753	0.60251	0.957379	0.859735	...	0.0	1
2	0.246914	0.045954	0.000000	0.0	0.000000	0.937500	0.698753	0.60251	0.957379	0.859735	...	0.0	1
3	0.283951	0.030704	0.000000	0.0	0.000000	0.937500	0.698753	0.60251	0.957379	0.859735	...	0.0	1
4	0.481481	0.062424	0.000000	0.0	0.000000	0.937500	0.698753	0.60251	0.957379	0.859735	...	0.0	1
...	...	...	...	...	...	...	...	...	...	...	...	...	
40770	0.691358	0.067914	0.000000	0.0	0.000000	0.479167	1.000000	0.00000	0.089322	0.000000	...	1.0	0
40771	0.358025	0.077877	0.000000	0.0	0.000000	0.479167	1.000000	0.00000	0.089322	0.000000	...	1.0	0
40772	0.481481	0.038430	0.018182	0.0	0.000000	0.479167	1.000000	0.00000	0.089322	0.000000	...	1.0	0
40773	0.333333	0.089874	0.000000	0.0	0.000000	0.479167	1.000000	0.00000	0.089322	0.000000	...	1.0	0
40774	0.703704	0.048597	0.036364	0.0	0.142857	0.479167	1.000000	0.00000	0.089322	0.000000	...	1.0	0

40775 rows × 59 columns

2.4 Transformations

We encoded the categorical features to numerical values used in the model to get better-predicted values. For this, we are going to use a one-hot encoding. To binarize these categorical variables, we apply hot encoding. One-hot encoding is done since the ML models can not use text-based features to train models.

We use pd.get_dummies as an encoding method since we want a fair and quantifiable distribution. General encoding gives [0, infinite] values that we do not wish to.

```
In [10]: transformed_data = pd.get_dummies(df, columns=cat_columns)

         transformed_data.head()
```

Out[10]:

	age	duration	campaign	pdays	previous	emp.var.rate	cons.price.idx	cons.conf.idx	euribor3m	nr.employed	...	day_of_week_fri	day_of_week_mon	day_of
0	56	261.0	1	0.0	0	1.1	93.994	-36.4	4.857	5191.0	...	0	1	
1	57	149.0	1	0.0	0	1.1	93.994	-36.4	4.857	5191.0	...	0	1	
2	37	226.0	1	0.0	0	1.1	93.994	-36.4	4.857	5191.0	...	0	1	
3	40	151.0	1	0.0	0	1.1	93.994	-36.4	4.857	5191.0	...	0	1	
4	56	307.0	1	0.0	0	1.1	93.994	-36.4	4.857	5191.0	...	0	1	

5 rows × 59 columns

2.5 Feature Selection

2.5.1 Filter Method

We used this method to create a hypothesis using the Chi-squared test; we set the alpha rate to 0.05 and ran the test on all the features. We were able to identify 15 independent features, and we have removed them.

2.5.2 Wrapper Method

We have used Recursive Feature Elimination as the wrapper method to select the best 40 features for the Random Forest model. Since wrappers are model specific, we will be using the selected features only while training the random forest model.

3. Model Selection and Evaluation

3.1 Evaluation Metrics

Given the nature of our problem, we use a confusion matrix and accuracy ratings as our evaluation metrics. This is how a customer's willingness to accept a bank's term deposit is classified. As a result, our model's number of correct predictions is all that matters to us (accuracy). Confusion matrixes are also the most appealing tool for predictive data analytics since they aid in visualizing model results. We developed new evaluation metrics without relying on pre-existing libraries. Confusion matrix, precision, recall, accuracy, and F-1 scores were used.

1. Accuracy: It is given by the set of labels predicted for a sample which exactly matches the corresponding set of labels in y_true.
2. Recall: It is the amount up to which the model can predict the outcome.
3. Precision: It is the level up to which the prediction made by the model is precise.
4. F1 Score: It is the amount of data tested for the predictions.

3.2 Models

We will use the Naive Bayes algorithm, Logistic Regression, ADABoost, and Random Forest, as this is a classification problem. We induce each of the mentioned models by using the training dataset and then

test the models one by one. We will utilize Accuracy, Recall, Precision, Confusion Matrix, and F1 score to evaluate the model's performance. Based on these metrics, we select the best-performing model.

3.3 Evaluation

3.3.1 Evaluation Settings and Sampling

The training portion of the data accounts for 77% of the total dataset, whereas the testing section accounts for 33%. Random sampling was used to construct these sections. We use identical training and testing datasets for all models by selecting the random state while splitting our data.

3.3.2 Hyperparameter Optimization

The Hyperparameter we use is the GridSearch method for logistic regression and random forest. We got the hyperparameters {'C': 100.0, 'penalty': 'l2', 'solver': 'lbfgs'} and {'criterion': 'entropy', 'max_depth': 'max_features': 'auto', 'n_estimators': 500} ,By using this we have improved the accuracy by 0.3% and 0.4% respectively.

3.3.3 Evaluation

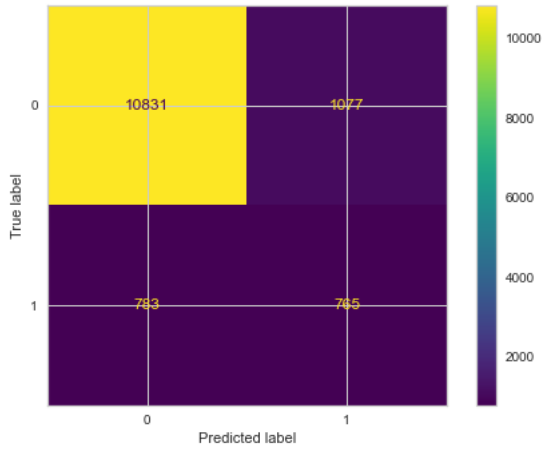
3.3.3.1 Naive Bayes

A Naive Bayes classifier naively assumes that each of the defining features in a domain is conditionally independent of all of the other descriptive features, given the state of the target feature. This algorithm uses Bayes' rule, which calculates the probability of an event related to previous knowledge of the variables concerning the event.

Implementation and Result

We import gaussianNB from sklearn and fit the function to train our model. After testing the model, we get overall accuracy of 0.86. The Precision for yes and no are 0.41 and 0.93, the Recall for yes and no are 0.49 and 0.91, and the F1 Score for yes and no are 0.45 and 0.92, respectively.

Now, we use another model, hoping to find a better accuracy.

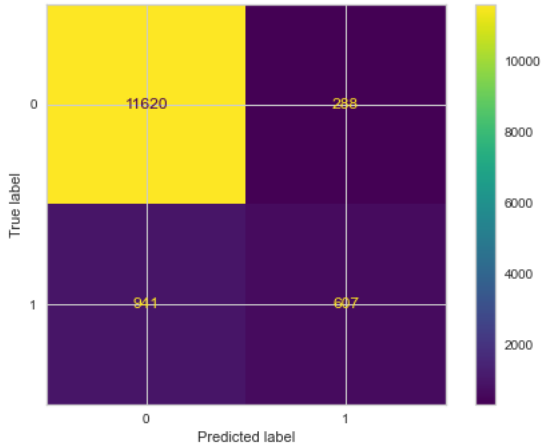


3.3.3.2 Logistic Regression

Logistic regression is a machine learning model that fits a linear decision boundary between the positive and negative samples. A line (Sigmoid function) predicts if the dependent variable is true or false based on the independent variables. In logistic regression, we have two possible values for the dependent variable, either 0 (false) or 1 (true). This model is very appropriate here as we are trying to predict whether the person accepts the offer of a bank term deposit or not. One advantage is figuring out which features are essential for predicting positive or negative.

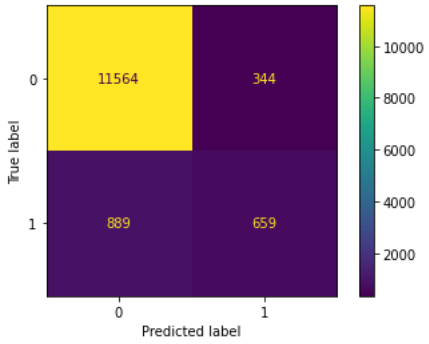
Implementation and Result

We import LogisticRegression from sklearn and build our model. After we tested the model on the testing data, we got an accuracy of 0.9086, better than the above model. The Precision for yes and no are 0.67 and 0.92, the Recall for yes and no are 0.39 and 0.97, and the F1 Score for yes and no is 0.49 and 0.95, respectively.



Hyperparameter finetuning

We use GridsearchCV to find out the best set of hyperparameters for our model. This hyperparameter tuning is done to arrive at an excellent value for the regularization constant, C. After doing this, we can see that the accuracy is the same. In contrast, the Recall for yes and the F1 Score for yes are increased to 0.42 and 0.51, respectively.

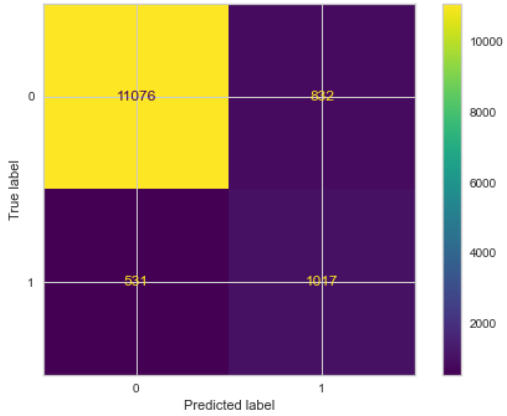


3.3.3.3 ADABoost

AdaBoost is one of the first boosting algorithms which uses an iterative approach to combine multiple weak classifiers into a single robust classifier. It is an ensemble learning method initially created to increase the efficiency of binary classifiers.

Implementation and Result

We import AdaBoostClassifier from sklearn and build our boosting model. After testing the model, we get overall accuracy of 0.898. The Precision for yes and no are 0.55 and 0.95, the Recall for yes and no are 0.65 and 0.93, and the F1 Score for yes and no is 0.59 and 0.94, respectively.
Compared to Logistic Regression, the accuracy is slightly less, but other metrics are somewhat more.

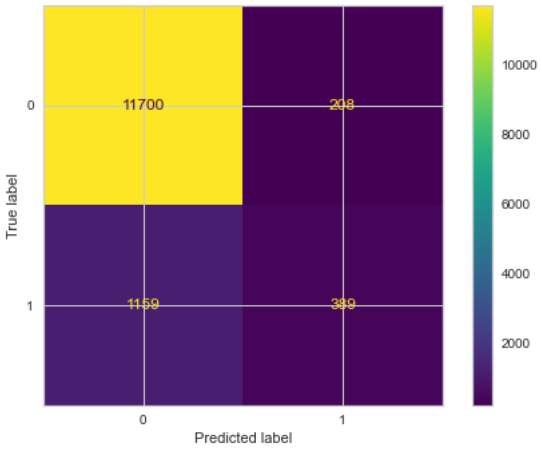


3.3.3.4 Support Vector Machine (SVM)

Support vector machine, also known as SVM, produces significant accuracy with less computation power. It is widely used in classification problems. It classifies data points into two classes at a time using a decision boundary, which is a hyperplane.

Implementation and Result

We import SVM from sklearn and build our boosting model. After testing the model, we get overall accuracy of 0.8984. The Precision for yes and no are 0.25 and 0.98, the Recall for yes and no are 0.65 and 0.90, and the F1 Score for yes and no are 0.36 and 0.94, respectively.

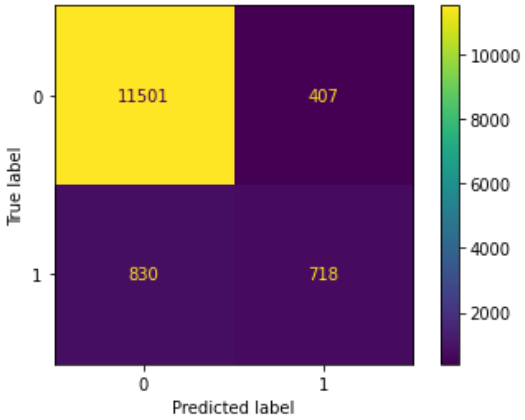


3.3.3.5 Random Forest

A random forest is a meta estimator that fits several decision tree classifiers on various sub-samples of the dataset and uses averaging to improve the predictive accuracy and control over-fitting. It tries to create an uncorrelated forest of trees whose prediction is more accurate than any individual tree by using the bagging and feature randomness method.

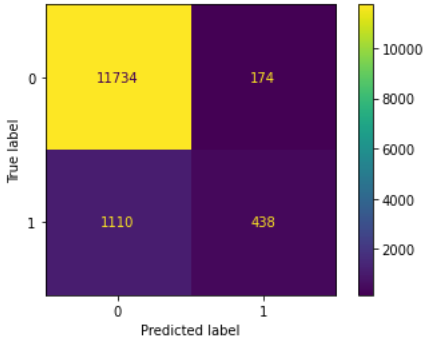
Implementation and Result

We import RandomForestClassifier from sklearn and fit the function to train our model. After testing the model, we get overall accuracy of 0.9080. The Precision for yes and no are 0.63 and 0.93, the Recall for yes and no are 0.46 and 0.96, and the F1 Score for yes and no is 0.53 and 0.94, respectively.



Hyperparameter finetuning

We use the same method as before, GridsearchCV, to find out the best set of hyperparameters for our model. After doing this, we can see that the accuracy is the same, whereas the Recall for yes, the F1 Score for yes, and the Precision for yes are slightly decreased. The Recall for no, the F1 Score for no, and the Precision for no are somewhat increased.



4 Results and Conclusion

The comparison table for all the models with their respective evaluation metrics can be seen below.

Model	Accuracy	Recall		Precision		F1 Score	
		Yes	No	Yes	No	Yes	No
Naïve Bayes	0.86	0.49	0.91	0.41	0.93	0.45	0.92
Logistic Regression	0.9086	0.39	0.97	0.67	0.92	0.49	0.95
Logistic Regression (Hyperparameter Finetuning)	0.9083	0.42	0.97	0.65	0.92	0.51	0.94
ADABoost	0.8987	0.65	0.93	0.55	0.95	0.59	0.94
SVM	0.8984	0.65	0.90	0.25	0.98	0.36	0.94
Random Forest	0.9080	0.46	0.96	0.63	0.93	0.53	0.94
Random Forest (Hyperparameter Finetuning)	0.9045	0.28	0.98	0.71	0.91	0.4	0.94

Out of all these models, the perfect model to select based on our calculations of evaluation metrics is Logistic Regression, with an accuracy of 0.9086. The next best model will be Random Forest, whose accuracy is 0.9080, almost like the previous model. The other best models are ADABoost and SVM, whose accuracy is below Logistic Regression.

Furthermore, the logistic model is more straightforward, making it less expensive and more efficient, especially since sample sizes would skyrocket if used in the field. The logistic model will demonstrate that it can suit our business needs while also relatively inexpensive and easy to duplicate.