Overview

In this assignment, we were provided a dataset of customers at a Portuguese bank that were targeted for a marketing campaign. The data is based on marketing campaigns conducted via phone cand the goal is to predict if the client will subscribe to a term deposit. The goal of the assignment is to build a model to predict if a client will subscribe to a term deposit.

Evaluation

For this exercise, I decided to use accuracy, specifically accuracy_score from sklearn.metrics, as the function to determine the metric to optimize for all models, comparing test holdback to predicted values from the models utilized. This is because I wanted to optimize the test data being correctly classified to cross check that I was not overfitting.

Data Understanding

I was prompted to first use **bank client data**. The **bank client data** included the customer's age, type of job, marital status, education level, credit default status, housing loan status, and personal loan status. I decided to add additional features to the **bank client data**. In **subsequent runs**, in addition to the bank client data, I added the marketing last contact month of year, the marketing last contact duration, the marketing number of contacts performed during this campaign, the marketing number of contacts performed before this campaign, the outcome of the previous marketing campaign, the quarterly employment variation rate, the monthly consumer price index, the monthly consumer confidence index, the euribor 3 month rate - daily indicator, and the number of unemployed quarterly indicator.

Data Preparation

For some of these data fields, the data type was “object” so I had to encode some of the fields to numerical values so that I could use them in my modeling using ***le.fit_transform()***.

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 41188 entries, 0 to 41187
Data columns (total 21 columns):
 #   Column          Non-Null Count  Dtype  
---  ------          --------------  -----  
 0   age             41188 non-null  int64  
 1   job             41188 non-null  object 
 2   marital         41188 non-null  object 
 3   education       41188 non-null  object 
 4   default         41188 non-null  object 
 5   housing         41188 non-null  object 
 6   loan            41188 non-null  object 
 7   contact         41188 non-null  object 
 8   month           41188 non-null  object 
 9   day_of_week     41188 non-null  object 
 10  duration        41188 non-null  int64  
 11  campaign        41188 non-null  int64  
 12  pdays           41188 non-null  int64  
 13  previous        41188 non-null  int64  
 14  poutcome        41188 non-null  object 
 15  emp.var.rate    41188 non-null  float64
 16  cons.price.idx  41188 non-null  float64
 17  cons.conf.idx   41188 non-null  float64
 18  euribor3m       41188 non-null  float64
 19  nr.employed     41188 non-null  float64
 20  y               41188 non-null  object 
dtypes: float64(5), int64(5), object(11)
memory usage: 6.6+ MB
```

For the **bank client data**, I encoded the type of job, marital status, education level, credit default status, housing loan status, and the personal loan status. For the **subsequent runs**, the marketing last contact month of year, and the outcome of the previous marketing campaign were also encoded. I also had to encode the outcome “y”, “did the client subscribe to a term deposit?”

Methodology

Once I had a solid understanding of the underlying dataset and had encoded applicable fields, I went about building the different models that were prompted for the exercise, namely K Nearest Neighbor, Logistic Regression, Decision Trees, and Support Vector Machines (‘rbf’, ‘linear’, ‘sigmoid’ and ‘poly’). For all runs of the SVM, I used GridSearchCV to find the optimal gamma and then based on the 1st rank in the “rank_test_score” array in the “cv_results_”, I did a subsequent run with the optimal gamma to measure timing of the run and accuracy.

When I added features for **subsequent runs**, I added additional GridSearchCV runs to see the optimal number of neighbors for KNN and the optional depth for the Decision Tree. In a similar manner as above, I then based on the 1st rank in the “rank_test_score” array in the “cv_results_”, I did a subsequent run with the optimal parameter (number of neighbors, depth, respectively) to measure timing of the run and accuracy. For both runs of the logistic regression model (in the **bank client data** and **subsequent runs**) I also iterated on the max number of runs for the logistic regression method to eliminate warnings.

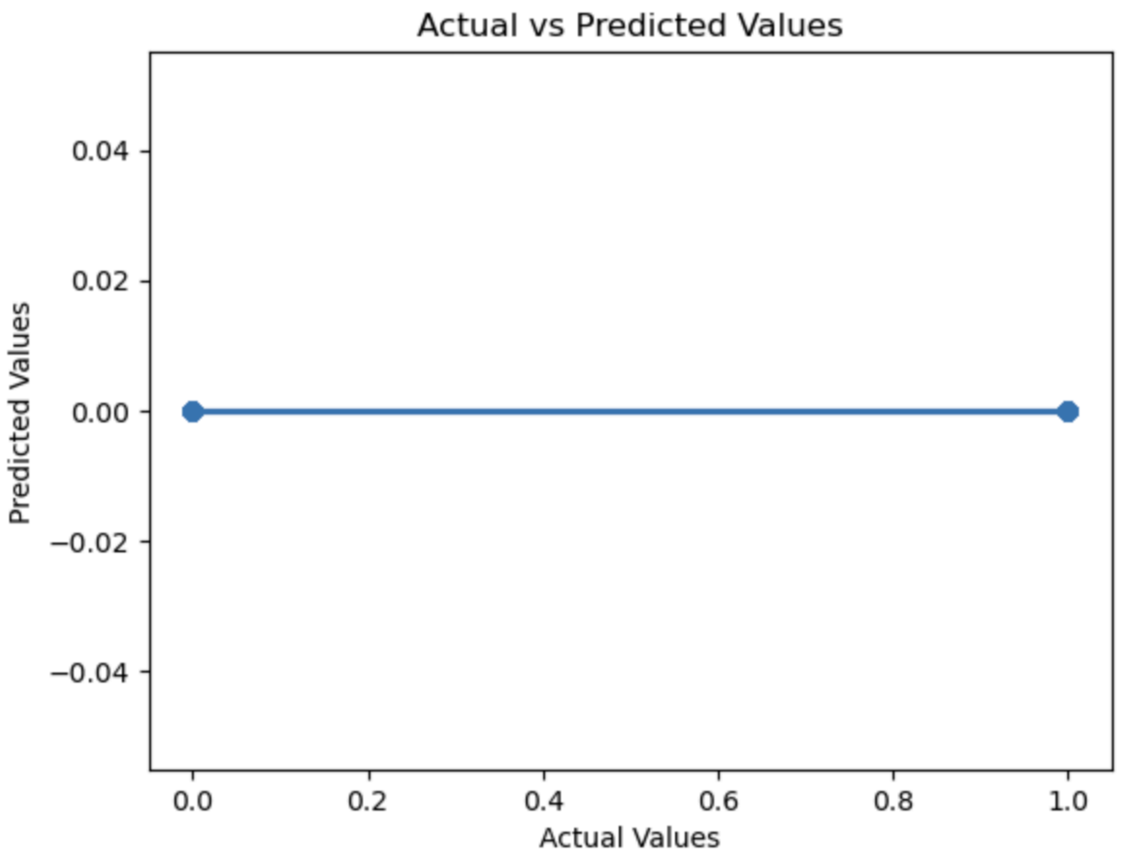
On the **bank client data** run, I populated both the training and the test accuracy into the dataframe per the prompts and then did the same for the **subsequent runs** after adding additional features to try and (successfully) improve the accuracy.

Summary Findings

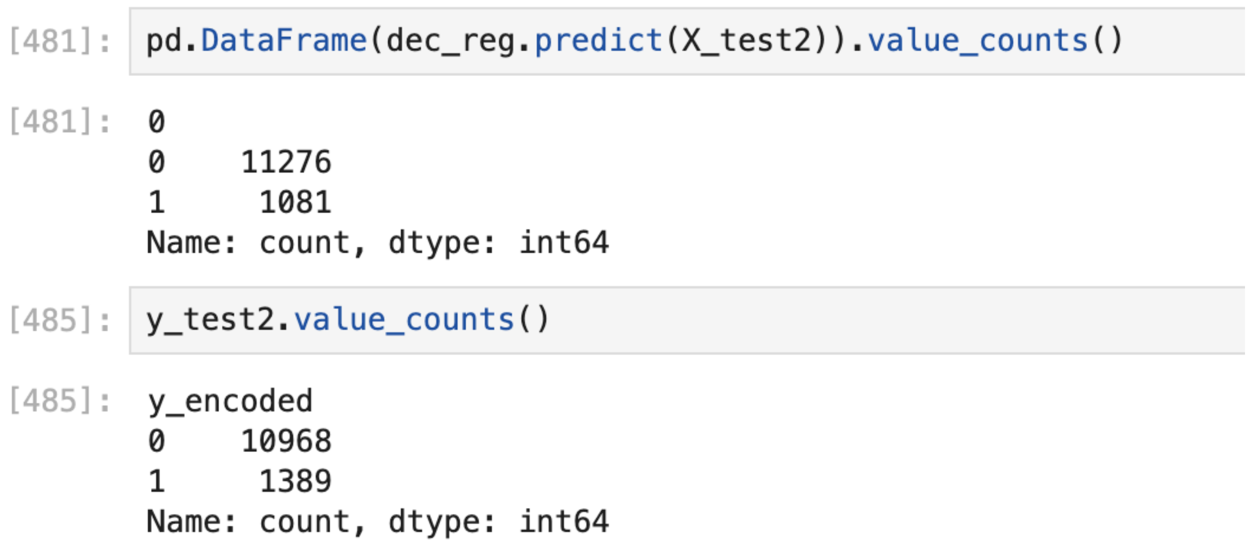
Overall, the best performing model I found using all the features I selected was a decision tree model with depth = 5. This model performed 3.1% better than the models that did not include the additional features beyond the **bank client data** in the training/test data set and 3.2% vs the default model of assuming a “no” with respect to a client subscribing to a term deposit based on the marketing campaign. In all scenarios, I could not get the poly SVC model to complete its run, likely due to the lopsidedness of the marketing campaign outcome which was a negative outcome 88.73% of the time.

For the model using only the **bank client data**, the predictions that come back for the X_train features are uninteresting, all optimal models predict “no” for the client subscribing to a term deposit.

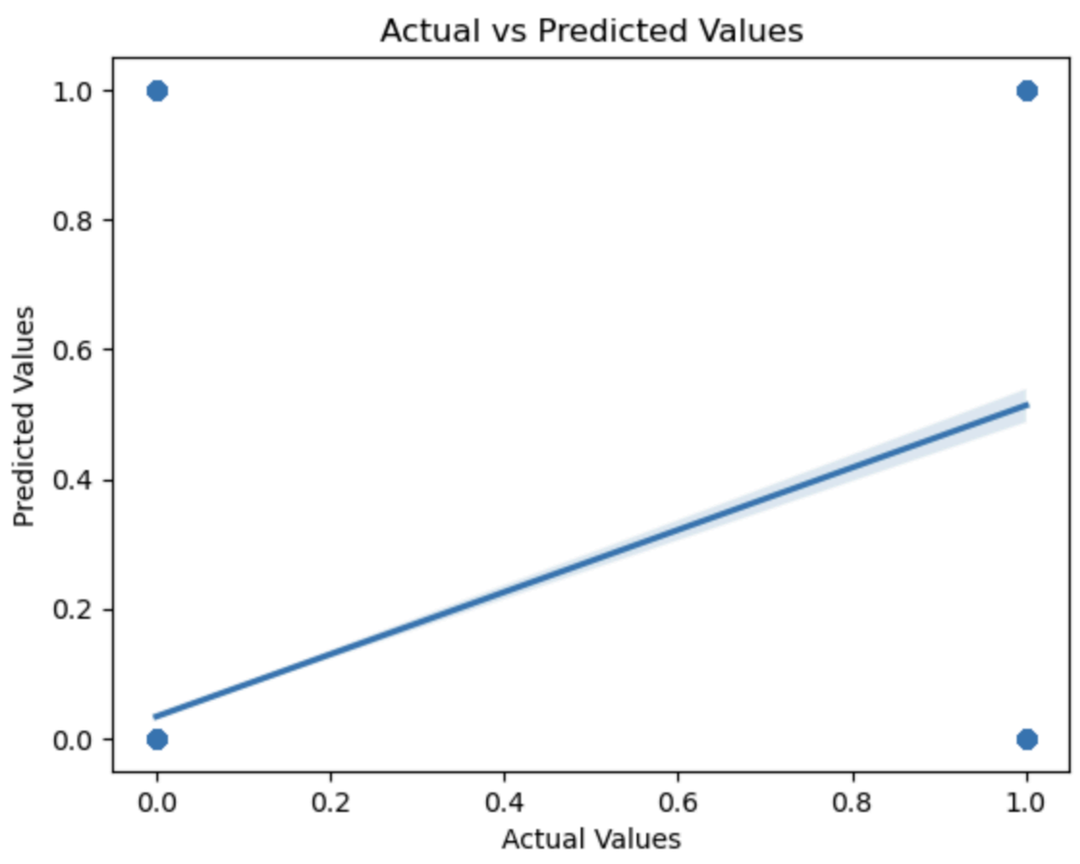
```
[434]: pd.DataFrame(log_reg.predict(X_test)).value_counts()

[434]: 0
       0    12357
       Name: count, dtype: int64

[430]: y_test.value_counts()

[430]: y_encoded
       0    10968
       1     1389
       Name: count, dtype: int64
```



For the **subsequent runs**, with the improved accuracy, I see the following for the X_train features using the optimal decision tree model wherein positive values are predicted for the test dataset.



```
[481]: pd.DataFrame(dec_reg.predict(X_test2)).value_counts()

[481]: 0
       0    11276
       1     1081
       Name: count, dtype: int64

[485]: y_test2.value_counts()

[485]: y_encoded
       0    10968
       1     1389
       Name: count, dtype: int64
```



Next Steps and Recommendations

In terms of actionability, utilizing the decision tree model of depth 5 will provide a useful approach to predicting the efficacy of the marketing related programs and I would advise the bank to utilize the model utilizing the following features

1. customer’s age
2. type of job
3. marital status
4. education level
5. credit default status
6. housing loan status
7. personal loan status
8. marketing last contact month of year
9. marketing last contact duration
10. marketing number of contacts performed during this campaign
11. marketing number of contacts performed before this campaign
12. outcome of the previous marketing campaign
13. quarterly employment variation rate
14. monthly consumer price index
15. monthly consumer confidence index
16. euribor 3 month rate - daily indicator

17. number of unemployed quarterly indicator

Resources

Link to notebook: · link to the notebook