**D214: Data Analytics Graduate Capstone**

# Predicting Term Deposits Using Logistic Regression and Random Forest Classifiers

**Andrew Mecchi**

**Masters of Data Analytics, Western Governors University**

The following report covers the Task Two Performance Assessment for D214 Data Analytics Graduate Capstone. This document is categorized by the questions defined in the rubric.

# Part I: Research Question

## A. Summarize the original real-data research question, justification for the research question, a description of the context, and a discussion of null and alternate hypotheses.

The ability to understand customer habits and tendencies is paramount for any business who desires to improve customer outreach strategy and construct predictive models. One such case, a banking institution wants to evaluate and predict successful term deposits based on a direct-marketing campaign. The bank implemented a telemarketing strategy to contact customers in the hope patrons would successfully complete a term deposit with their institution. Throughout the years-long campaign, the bank collected data from customers totalling 45,211 records and included 17 features. The purpose of this study is to create a predictive model capable of accurately forecasting customers who will subscribe to term deposits. A successful model can be employed by banks that are looking to predict if customers will subscribe to term deposits with their branch. Therefore, the following research question is proposed:

**Is it possible to predict term deposits for customers using logistic regression and a random forest classifier from imbalanced data with an F1 score greater than 0.3 or 30%?**

This study will employ a logistic regression classification model and a random forest classifier to project term deposits. With the proposed research question, there are two hypotheses that will be tested. The null hypothesis states that a predictive model cannot classify customer term deposits with statistical significance. Meaning, through data cleaning, feature engineering, and data modeling, term deposits cannot be accurately predicted from the given data with an F1 score greater than 0.30 or 30%. The alternate hypothesis suggests a predictive model can successfully identify term deposits from the given data with an F1 statistic that exceeds 0.30 or 30%. A predictive model that achieves an F1 score greater than 0.30 will not only assist banks with the ability to identify future clients' term deposits, but will also inform them of the most significant feature(s) contributing to term deposit success. Having an accurate model benefits banks in identifying clients who are likely to subscribe to term deposits with their branch, thus, improving business and promoting growth. Additionally, the institutions can enhance their marketing strategies from the insights gained through the identification of influential feature(s) contributing to the model. Both logistic regression and random forest classification algorithms have the ability to evaluate the variables and make classifications based on the data available.

Logistic regression is a classification technique which utilizes a statistical method for analyzing a dataset in which there are one or more independent variables that determine an outcome with the best fitting model able to describe the relationship between the dependent and the independent variable(s) (Raj, 2020). The random forest classifier is a supervised machine learning algorithm used for classification by implementing the creation of a set of decision trees from a randomly selected subset of training data and collects the votes from the different decision trees to make the final classification (Geeks for Geeks, n.d.). Jiang et. al. (2022) constructed a logistic regression classification model capable of predicting successful term deposits from a bank's telemarketing campaign. The researchers identified predictor variables

related to term deposits using feature engineering, data cleaning, and created dummy variables for the categorical data previous to implementing their logistic regression classification model. Similarly, Muslim et. al. (2021) employed a random forest machine learning classifier for long-term bank deposit investors and affirmed the prowess of the algorithm by predicting term deposits using a random forest classifier and SMOTE (synthetic minority oversampling technique). After thorough data cleaning and encoding of categorical variables, an accurate predictive model should be possible with logistic regression and random forest classifiers.

## Part II: Data Collection

### B. Report on the data-collection process by describing the relevant data, discuss one advantage and one disadvantage of the methodology used, and discuss any challenges.

The collection of data required for a successful analysis already exists and is available for public download via University of California Irvine (UCI) Machine Learning Repository and can be found through the following link, Predict Term Deposits. The dataset's provenance is sourced from Moro et. el. (2014) and their study assessing the success of bank telemarketing decision support systems with data collected by a Portuguese bank from 2008 to 2013. Previous to any data treatment, the dataset contains 45,211 rows and 17 columns with no data sparsity. The dataset includes the following variables; age, job, marital, education, default (renamed credit_default), balance, housing, loan, contact, day, month, duration, campaign, pdays (renamed days_since_contact), previous (renamed previous_campaign_contacts), poutcome (renamed previous_outcome), and the target variable, y (renamed deposit_success). The description and role of each variable for this analysis is broken down below in Table 1.

**Table 1: Composition of Term Deposit Dataset Analyzed**

| Variable Name | Variable Type | Data Type |
|---|---|---|
| deposit_success | Categorical | Dependent Variable |
| age | Continuous | Independent Variable |
| job | Categorical | Independent Variable |
| marital | Categorical | Independent Variable |
| education | Categorical | Independent Variable |
| credit_default | Categorical | Independent Variable |
| balance | Continuous | Independent Variable |
| housing | Categorical | Independent Variable |
| loan | Categorical | Independent Variable |

| | | |
|---|---|---|
| contact | Continuous | Independent Variable |
| day | Continuous | Independent Variable |
| month | Categorical | Independent Variable |
| duration | Continuous | Independent Variable |
| campaign | Continuous | Independent Variable |
| days_since_contact | Continuous → re-expressed as Categorical | Independent Variable |
| previous_campaing_contacts | Continuous | Independent Variable |
| previous_campaign_outcome | Categorical | Independent Variable |

Given the dataset, it is also important to scrutinize the advantages and disadvantages of the data availability, limitations, and delimitations. The greatest strength of the data set lies in its convenience, completeness, and contains no data sparsity. The number of records without missing data is a great advantage for the analysis as there is high quality data and does not require imputations for any missing values. However, the data is limited by the accuracy of the bank's recordkeeping and ability to correctly collect the information provided within the dataset. Additionally, the outcome of term deposits is assumed as a direct result of the marketing campaign. Furthermore, the feature column duration contains values of zero, therefore, these records cannot be used in the model because marketing success cannot be attributed to a direct-marketing strategy when contact was never made. To address the challenge of potentially confounding records, instances where duration contained values of zero were removed from the analysis.

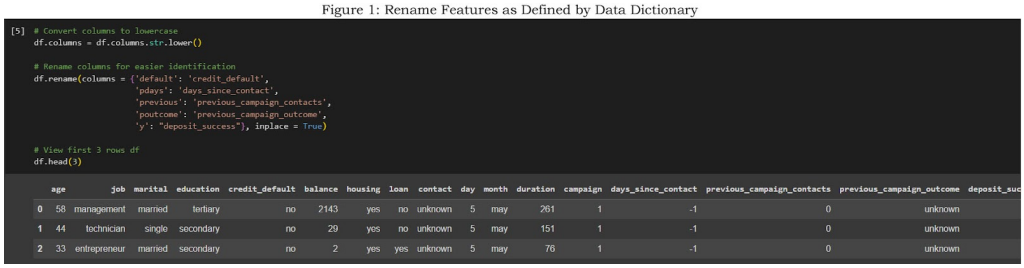
The greatest disadvantage of the data is the inherent imbalance of term deposit success. Without any data treatment there is a term deposit success rate of approximately 11.7%, meaning, the representation of positive success cases is in the minority of data. The ability to acquire more data to reduce the imbalance is not possible, therefore, to overcome the challenge of classification imbalance, synthetic minority over-sampling technique or SMOTE is used to resolve the under-represented minority class. Lastly, the dataset is delimited through feature manipulation and is treated in a manner that removes outliers to confine the model to the most statistically significant range of variables. Outliers can be the cause of data corruption, measurement errors, and the presence of outliers in a dataset impacts the model to a great extent, thus, handling outliers is an essential component of the feature engineering pipeline (Kumar, 2021). In summation, the dataset presents great advantages and disadvantages, but through careful data preprocessing steps, feature engineering, and addressing the classification imbalance using SMOTE, an accurate predictive model should be possible through logistic regression and random forest classifiers.

# Part III: Data Extraction and Preparation

## C. Describe your data-extraction and preparation process, explain the tools and techniques used, justify them, and discuss an advantage and disadvantage.

The goal of this analysis is to create an accurate model capable of predicting term deposit success based on data collected by a bank during a direct-marketing campaign. To ensure an optimal model is created, the data will undergo cleaning, feature engineering, and removal of statistical outliers. The first treatment was to rename columns to better understand the variables in context of the analysis (Fig. 1). The data dictionary was used to identify the following features, default, pdays, previous, poutcome, and y. Default categorized patrons in credit default and was renamed “credit_default.” Pdays quantified the number of days since a customer was last contacted from a previous campaign and was renamed, “days_since_contact.” Both previous and poutcome described the number of previous campaign contacts and their respective outcomes which were renamed accordingly (previous_campaign_contacts, previous_campaign_outcome). Lastly, y, the target variable, was renamed deposit_success to better represent the feature in visuals and analysis.

Figure 1: Rename Features as Defined by Data Dictionary



```
[5] # Convert columns to lowercase
    df.columns = df.columns.str.lower()

    # Rename columns for easier identification
    df.rename(columns = {'default': 'credit_default',
                         'pdays': 'days_since_contact',
                         'previous': 'previous_campaign_contacts',
                         'poutcome': 'previous_campaign_outcome',
                         'y': "deposit_success"}, inplace = True)

    # View first 3 rows df
    df.head(3)
```

| | age | job | marital | education | credit_default | balance | housing | loan | contact | day | month | duration | campaign | days_since_contact | previous_campaign_contacts | previous_campaign_outcome | deposit_suc |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 58 | management | married | tertiary | no | 2143 | yes | no | unknown | 5 | may | 261 | 1 | -1 | 0 | unknown | |
| 1 | 44 | technician | single | secondary | no | 29 | yes | no | unknown | 5 | may | 151 | 1 | -1 | 0 | unknown | |
| 2 | 33 | entrepreneur | married | secondary | no | 2 | yes | yes | unknown | 5 | may | 76 | 1 | -1 | 0 | unknown | |

Following the relabelling of the variables, the data was explored to identify any missing/null values and duplicate records, an inquiry which returned no missing values or duplicate entries. Next came the treatment for the variable duration because values for this feature must exceed zero as the results were based on marketing campaign success. Thus, in order for the effect of a marketing campaign to quantify a result it was determined that the customer must be contacted. Therefore, duration records with a value of zero were identified and removed from the dataframe through the application of a filter to retain all records with duration values greater than or equal to one (Fig. 2). With the necessary preparations and variables relabeled, data distributions of quantitative features were visualized using histograms (Fig. 3), box plots (Fig. 4), and pair plots (Fig 5.) After observing the data visualizations, the analysis process proceeded to feature engineering and data cleaning.

Figure 2: Removal of Duration Records With Value of Zero and View Summary Statistics

```
[10] # Treat duration column --- duration is defined as time elapsed when contacted, if not contacted and no duration
     # View number of duration calls with values of 0
     df[df['duration'] == 0]
```

| | age | job | marital | education | credit_default | balance | housing | loan | contact | day | month | duration | campaign | days_since_contact | previous_campaign_contacts | previous_campaign_outcome | deposit_success |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 6424 | 53 | management | married | primary | no | 351 | yes | no | unknown | 27 | may | 0 | 4 | -1 | 0 | unknown | no |
| 22937 | 35 | technician | married | secondary | no | 5535 | no | no | cellular | 26 | aug | 0 | 15 | -1 | 0 | unknown | no |
| 36425 | 31 | entrepreneur | married | secondary | no | 162 | yes | yes | cellular | 11 | may | 0 | 2 | -1 | 0 | unknown | no |

```
[11] # Duration filter found 3 clients with no duration/no previous campaign contact/no deposits
     # proceed to filter df where duration is 0

     df = df[df['duration'] >= 1]
```

```
[12] # Primary view of summary statistics - numeric columns
     df.describe()
```

| | age | balance | day | duration | campaign | days_since_contact | previous_campaign_contacts |
|---|---|---|---|---|---|---|---|
| count | 45208.000 | 45208.000 | 45208.000 | 45208.000 | 45208.000 | 45208.000 | 45208.000 |
| mean | 40.936 | 1362.229 | 15.806 | 258.180 | 2.764 | 40.201 | 0.580 |
| std | 10.619 | 3044.795 | 8.322 | 257.528 | 3.098 | 100.132 | 2.304 |
| min | 18.000 | -8019.000 | 1.000 | 1.000 | 1.000 | -1.000 | 0.000 |
| 25% | 33.000 | 72.000 | 8.000 | 103.000 | 1.000 | -1.000 | 0.000 |
| 50% | 39.000 | 448.000 | 16.000 | 180.000 | 2.000 | -1.000 | 0.000 |
| 75% | 48.000 | 1428.000 | 21.000 | 319.000 | 3.000 | -1.000 | 0.000 |
| max | 95.000 | 102127.000 | 31.000 | 4918.000 | 63.000 | 871.000 | 275.000 |

Figure 3: Histograms and Distribution of Data

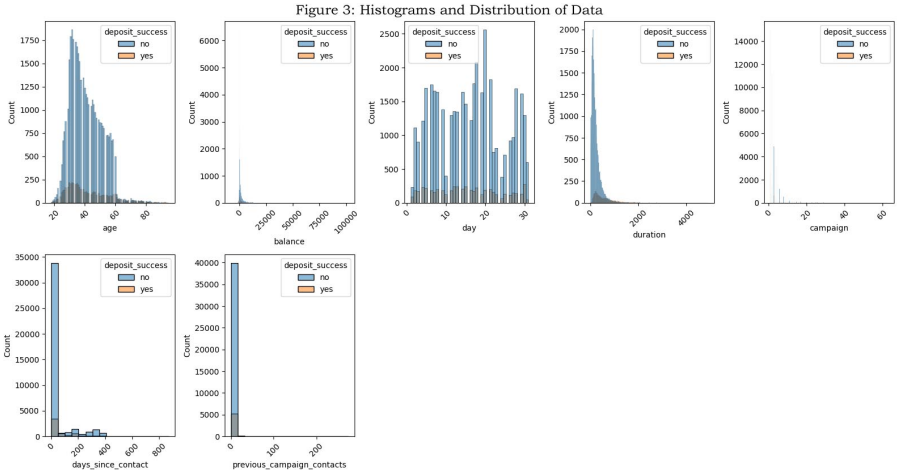



Figure 4: Box Plots and Distribution of Data

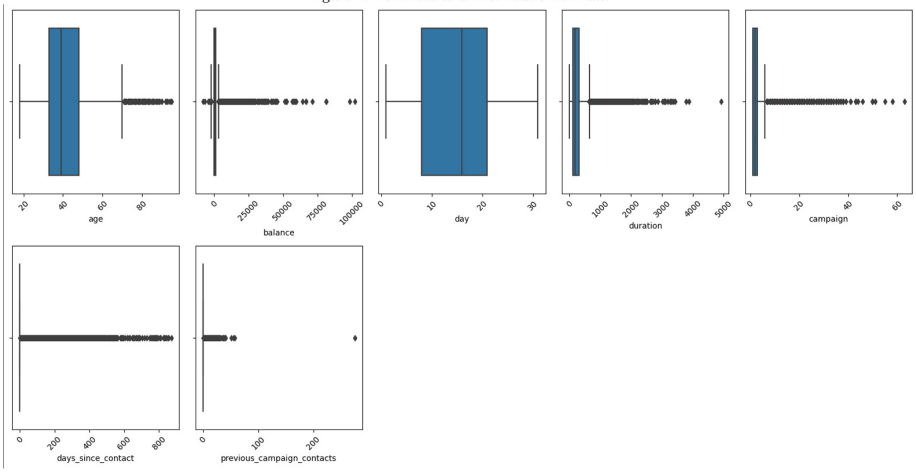
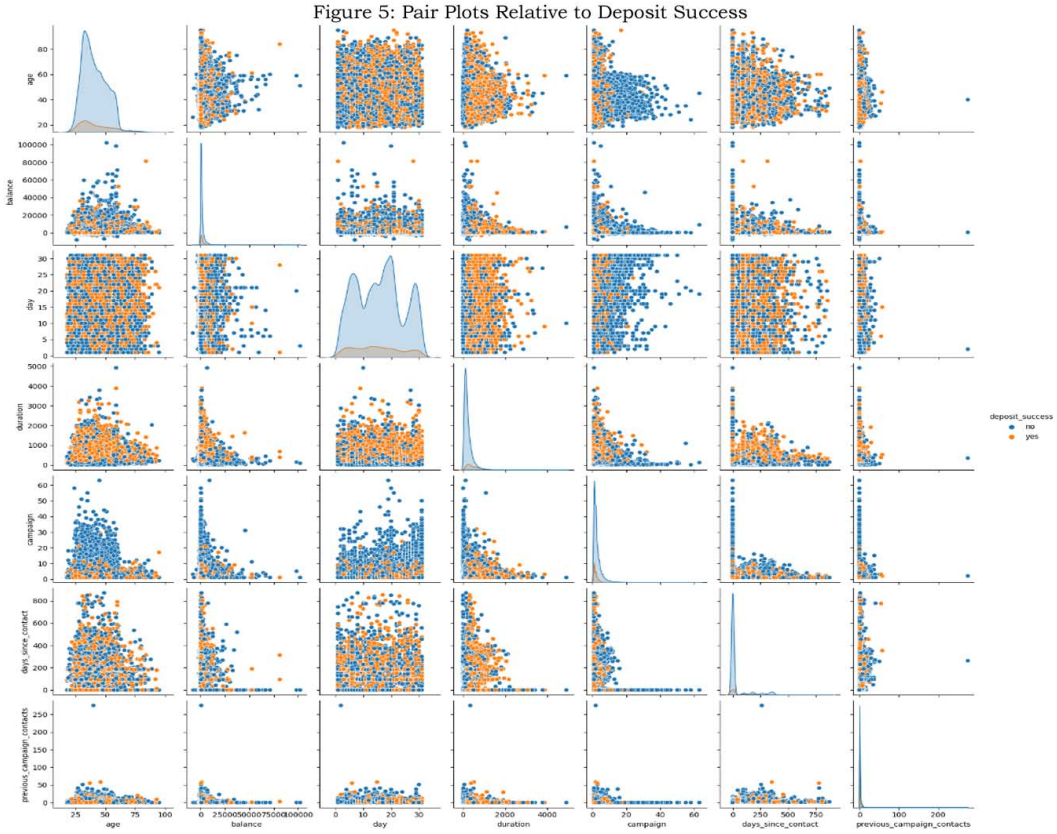
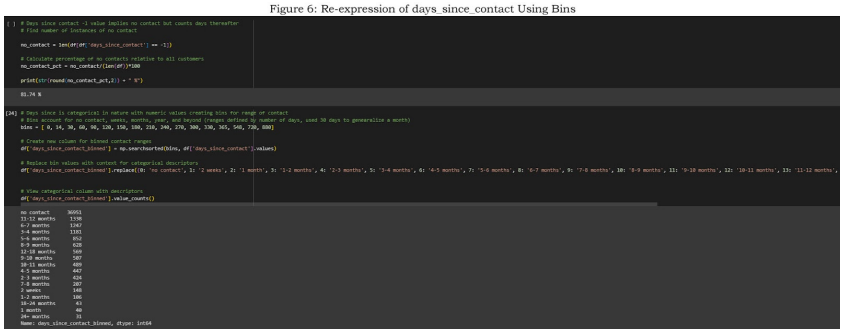


Figure 5: Pair Plots Relative to Deposit Success




After distributions were visualized, it was noted that the variable days_since_contact needed to be re-expressed as categorical after noticing a substantial skew towards -1. Referencing the data dictionary, the feature contained values relative to the number of days since contact of the bank's previous campaign. However, to denote patrons who were not contacted, a value of -1 was applied to those customers. The negative value for these customers drastically affects the summary statistics as approximately 82% of patrons were not previously contacted, therefore, the decision was made to re-express the variable as categorical using bins. The values of -1 were categorized as, "no contact" while bins were created with ranges in numbers of days; defined in terms of weeks and number of months (Fig. 6). With the confirmation of no missing or duplicate records, removal of zero-value duration records, and re-expression of days_since_contact; the data has been prepared for the data cleaning, outlier detection, and removal.

Figure 6: Re-expression of days_since_contact Using Bins



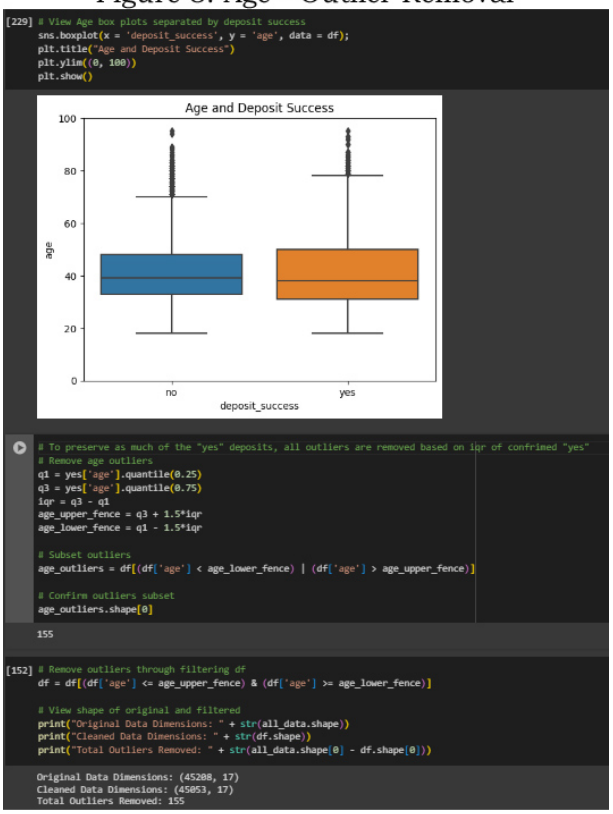
Of the two predictive model algorithms, logistic regression can be sensitive to outliers, therefore, the data must be treated for outlier detection and subsequent extrication. Having observed the skewed data distributions found in the histograms (Fig. 1) and box plots (Fig. 2), the interquartile range method was used for treatment of outliers. The interquartile range method is robust against outlier influence and computes the measure of spread of the interquartile range (IQR), or, the middle half of the data (Frost, 2023). The interquartile range method identifies outliers as data points that fall beyond the upper and lower limits defined by Q3 + 1.5*IQR (upper limit) and Q1 - 1.5*IQR (lower limit); where Q1 and Q3 are the first and third quartile and IQR is the difference between the third and first quartile. The advantage of using the interquartile method is the versatility of its application, robustness, and ability to remove outliers regardless of data distribution (normal, skewed, etc.). The disadvantage of using such an approach could significantly limit the amount of data being imputed into the model downstream. In other words, if the data contains significant quantities of outliers, the model will have less data points from which to learn. When applying the interquartile method to the term deposit data, preserving the positive cases remained paramount given the imbalance of term deposit success. Therefore, the outliers were detected using the IQR defined by the ranges of values from the records where term deposits were successful. This was done by splitting the main dataframe into two dataframes, one where term deposits were successful and the other where term deposits were unsuccessful (Fig. 7).

Figure 7: Isolating Term Deposit Success for IQR Method

```
# Want to preserve as much of the minority class as possible, treat outliers based on values of yes summary statistics
# Isolate Noes
no = df[df['deposit_success'] == 'no']

# Isolate Yeses
yes = df[df['deposit_success'] == 'yes']
```

The upper and lower fences were determined by the IQR calculations applied to the yes dataframe and then the identified outliers would be removed from the main dataframe and placed in a corresponding outlier dataframe (process repeated for each successive variable). These calculations were applied to the following features with the supporting code found in the images below; age (Fig. 8), balance (Fig. 9), duration (Fig. 10), campaign (Fig. 11), and previous_campaign_contacts (Fig. 12).

Figure 8: Age - Outlier Removal



```
[229] # View Age box plots separated by deposit success
      sns.boxplot(x = 'deposit_success', y = 'age', data = df);
      plt.title("Age and Deposit Success")
      plt.ylim((0, 100))
      plt.show()
```



```
# To preserve as much of the "yes" deposits, all outliers are removed based on iqr of confrimed "yes"
# Remove age outliers
q1 = yes['age'].quantile(0.25)
q3 = yes['age'].quantile(0.75)
iqr = q3 - q1
age_upper_fence = q3 + 1.5*iqr
age_lower_fence = q1 - 1.5*iqr

# Subset outliers
age_outliers = df[(df['age'] < age_lower_fence) | (df['age'] > age_upper_fence)]

# Confirm outliers subset
age_outliers.shape[0]
```

```
155
```

```
[152] # Remove outliers through filtering df
      df = df[(df['age'] <= age_upper_fence) & (df['age'] >= age_lower_fence)]

      # View shape of original and filtered
      print("Original Data Dimensions: " + str(all_data.shape))
      print("Cleaned Data Dimensions: " + str(df.shape))
      print("Total Outliers Removed: " + str(all_data.shape[0] - df.shape[0]))
```

```
Original Data Dimensions: (45208, 17)
Cleaned Data Dimensions: (45053, 17)
Total Outliers Removed: 155
```

Figure 9: Balance - Outlier Removal

```
[273] # View Balance box plots separated by deposit success
      sns.boxplot(x = 'deposit_success', y = 'balance', data = df);
      plt.ylim((-5000, 10000))
      plt.title("Balance and Deposit Success")
      plt.show()
```



```
[274] # To preserve as much of the "yes" deposits, all outliers are removed based on iqr of confrimed "yes"
      # Remove balance outliers
      q1 = yes['balance'].quantile(0.25)
      q3 = yes['balance'].quantile(0.75)
      iqr = q3 - q1
      balance_upper_fence = q3 + 1.5*iqr
      balance_lower_fence = q1 - 1.5*iqr

      # Subset outliers
      balance_outliers = df[(df['balance'] < balance_lower_fence) | (df['balance'] > balance_upper_fence)]

      balance_outliers.shape[0]
```

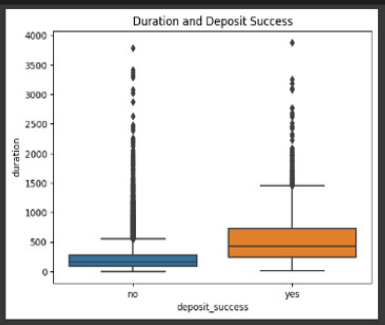
```
2759
```

```
[275] # Remove outliers through filtering df
      df = df[(df['balance'] <= balance_upper_fence) & (df['balance'] >= balance_lower_fence)]

      # View shape of original and filtered
      print("Original Data Dimensions: " + str(all_data.shape))
      print("Cleaned Data Dimensions: " + str(df.shape))
      print("Total Outliers Removed: " + str(all_data.shape[0] - df.shape[0]))
```

```
Original Data Dimensions: (45208, 17)
Cleaned Data Dimensions: (42294, 17)
Total Outliers Removed: 2914
```

Figure 10: Duration - Outlier Removal

```
[279] # View Balance box plots separated by deposit success
      sns.boxplot(x = 'deposit_success', y = 'duration', data = df);
      plt.title("Duration and Deposit Success")
      plt.show()
```

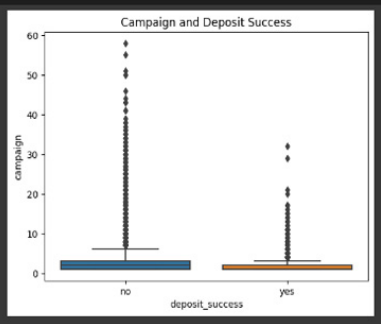



```
[280] # To preserve as much of the "yes" deposits, all outliers are removed based on iqr of confrimed "yes"
      # Remove duration outliers
      q1 = yes['duration'].quantile(0.25)
      q3 = yes['duration'].quantile(0.75)
      iqr = q3 - q1
      duration_upper_fence = q3 + 1.5*iqr
      duration_lower_fence = q1 - 1.5*iqr

      # Subset outliers
      duration_outliers = df[(df['duration'] < duration_lower_fence) | (df['duration'] > duration_upper_fence)]

      duration_outliers.shape[0]

      244
```

```
[281] # Remove outliers through filtering df
      df = df[(df['duration'] <= duration_upper_fence) & (df['duration'] >= duration_lower_fence)]

      # View shape of original and filtered
      print("Original Data Dimensions: " + str(all_data.shape))
      print("Cleaned Data Dimensions: " + str(df.shape))
      print("Total Outliers Removed: " + str(all_data.shape[0] - df.shape[0]))

      Original Data Dimensions: (45208, 17)
      Cleaned Data Dimensions: (42050, 17)
      Total Outliers Removed: 3158
```

Figure 11: Campaign - Outlier Removal

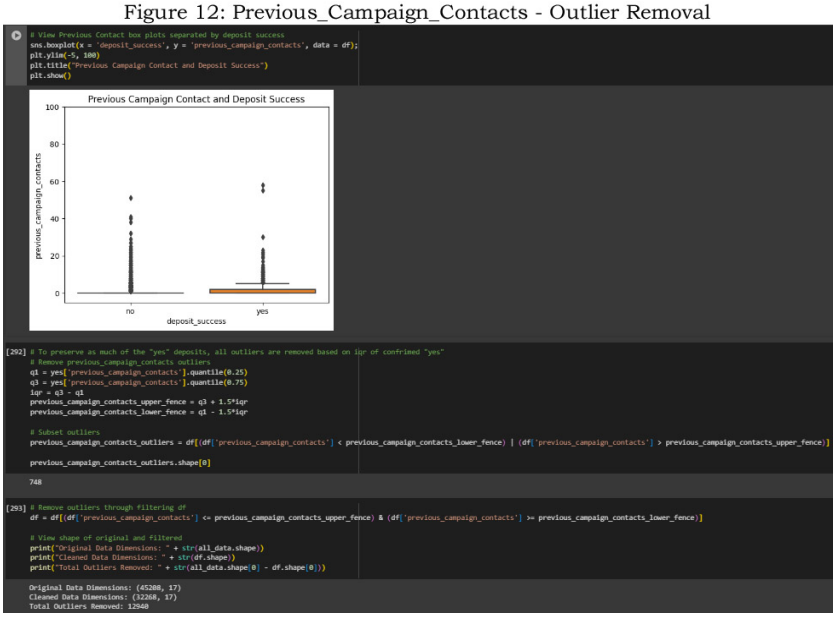
```
[285] # View Campaign box plots separated by deposit success
      sns.boxplot(x = 'deposit_success', y = 'campaign', data = df);
      plt.title("Campaign and Deposit Success")
      plt.show()
```




```
[286] # To preserve as much of the "yes" deposits, all outliers are removed based on iqr of confrimed "yes"
      # Remove campaign outliers
      q1 = yes['campaign'].quantile(0.25)
      q3 = yes['campaign'].quantile(0.75)
      iqr = q3 - q1
      campaign_upper_fence = q3 + 1.5*iqr
      campaign_lower_fence = q1 - 1.5*iqr


      # Subset outliers
      campaign_outliers = df[(df['campaign'] < campaign_lower_fence) | (df['campaign'] > campaign_upper_fence)]

      campaign_outliers.shape[0]

      9034
```

```
[287] # Remove outliers through filtering df
      df = df[(df['campaign'] <= campaign_upper_fence) & (df['campaign'] >= campaign_lower_fence)]

      # View shape of original and filtered
      print("Original Data Dimensions: " + str(all_data.shape))
      print("Cleaned Data Dimensions: " + str(df.shape))
      print("Total Outliers Removed: " + str(all_data.shape[0] - df.shape[0]))

      Original Data Dimensions: (45208, 17)
      Cleaned Data Dimensions: (33016, 17)
      Total Outliers Removed: 12192
```

Figure 12: Previous_Campaign_Contacts - Outlier Removal




```
# View Previous Contact box plots separated by deposit success
sns.boxplot(x = 'deposit_success', y = 'previous_campaign_contacts', data = df);
plt.ylim(-5, 100)
plt.title("Previous Campaign Contact and Deposit Success")
plt.show()
```

```
[292] # To preserve as much of the "yes" deposits, all outliers are removed based on iqr of confirmed "yes"
      # Remove previous_campaign_contacts outliers
      q1 = yes['previous_campaign_contacts'].quantile(0.25)
      q3 = yes['previous_campaign_contacts'].quantile(0.75)
      iqr = q3 - q1
      previous_campaign_contacts_upper_fence = q3 + 1.5*iqr
      previous_campaign_contacts_lower_fence = q1 - 1.5*iqr

      # Subset outliers
      previous_campaign_contacts_outliers = df[(df['previous_campaign_contacts'] < previous_campaign_contacts_lower_fence) | (df['previous_campaign_contacts'] > previous_campaign_contacts_upper_fence)]

      previous_campaign_contacts_outliers.shape[0]

      748

[293] # Remove outliers through filtering df
      df = df[(df['previous_campaign_contacts'] <= previous_campaign_contacts_upper_fence) & (df['previous_campaign_contacts'] >= previous_campaign_contacts_lower_fence)]

      # View shape of original and filtered
      print("Original Data Dimensions: " + str(all_data.shape))
      print("Cleaned Data Dimensions: " + str(df.shape))
      print("Total Outliers Removed: " + str(all_data.shape[0] - df.shape[0]))

      Original Data Dimensions: (45208, 17)
      Cleaned Data Dimensions: (32268, 17)
      Total Outliers Removed: 12940
```

Following the completion of outlier removal, each respective outlier dataframe (age_outliers, balance_outliers, duration_outliers, campaign_outliers, previous_campaign_outcome_outliers) was concatenated into a singular outlier dataframe. There were a total of 12,940 outliers removed which represents approximately 28.6% of the records from the original dataframe. The overall percentage of successful term deposits remained relatively unchanged with a value of 11.8% success (increase of 0.1%) (Fig. 13).

Figure 13: Post Outlier Removal Analysis

```
[299] # Update yeses and noes after days since contact removal
      nos = df[df['deposit_success'] == 'no']
      yes = df[df['deposit_success'] == 'yes']

[300] # Percentage of total yes After cleaning
      print("Percentage of all Successful Deposits After Cleaning: " + str(round(len(yes)/len(df),3)))

      Percentage of all Successful Deposits After Cleaning: 0.118

[301] # Total outliers removed
      print("Total Outliers Removed: " + str(all_data.shape[0] - df.shape[0]))

      # Percent of all data removed
      print("Percent of Data Removed: " + str((all_data.shape[0] - df.shape[0])/all_data.shape[0]))

      Total Outliers Removed: 12940
      Percent of Data Removed: 0.2862325252167758

      outliers = pd.concat([age_outliers, balance_outliers, duration_outliers, campaign_outliers, previous_campaign_contacts_outliers], ignore_index = True)

      outliers.info()

      <class 'pandas.core.frame.DataFrame'>
      RangeIndex: 12940 entries, 0 to 12939
      Data columns (total 17 columns):
       #   Column                     Non-Null Count  Dtype
      ---  ------                     --------------  -----
       0   age                        12940 non-null  int64
       1   job                        12940 non-null  object
       2   marital                    12940 non-null  object
       3   education                  12940 non-null  object
       4   credit_default             12940 non-null  object
       5   balance                    12940 non-null  int64
       6   housing                    12940 non-null  object
       7   loan                       12940 non-null  object
       8   contact                    12940 non-null  object
       9   day                        12940 non-null  int64
       10  month                      12940 non-null  object
       11  duration                   12940 non-null  int64
       12  campaign                   12940 non-null  int64
       13  previous_campaign_contacts 12940 non-null  int64
       14  previous_campaign_outcome  12940 non-null  object
       15  deposit_success            12940 non-null  object
       16  days_since_contact_binned  12940 non-null  category
      dtypes: category(1), int64(6), object(10)
      memory usage: 1.6+ MB
```

After the removal of statistical outliers, the cleaned data is ready to progress to the next stage. A Shapiro-Wilk test was used to examine the normality of data distribution, but found no normal distributions among the features (Fig. 14). The analysis will undergo logistic regression and

random forest classification models, both of which a normal distribution is not required, therefore no additional transformations of data were necessary. As stated by Roy et. al. (2015), logistic regression does not assume a linear relationship between the dependent and independent variables and the independent variables need neither be normally distributed, nor linearly related. Similarly, in a random forest, each node is not comparing feature values, it is simply splitting a sorted list that requires absolute values for branching and is based on partitioning the data to make predictions, therefore, it does not require normalization (Arya, 2022). The preparation has reached its final steps where categorical data will be treated, train and test sets defined, data scaled, and SMOTE applied.

Figure 14: Results of Shapiro-Wilk Test for Normality

```
[184] # Shapiro Wilk test for normality
      for col in df.columns:
        if df[col].dtype == 'int64':
          print(f"Shapiro-Wilk test for {col}: {shapiro(df[col])}")

      Shapiro-Wilk test for age: ShapiroResult(statistic=0.9642646312713623, pvalue=0.0)
      Shapiro-Wilk test for balance: ShapiroResult(statistic=0.817630410194397, pvalue=0.0)
      Shapiro-Wilk test for day: ShapiroResult(statistic=0.9577406644821167, pvalue=0.0)
      Shapiro-Wilk test for duration: ShapiroResult(statistic=0.8036201596260071, pvalue=0.0)
      Shapiro-Wilk test for campaign: ShapiroResult(statistic=0.7605543732643127, pvalue=0.0)
      Shapiro-Wilk test for previous_campaign_contacts: ShapiroResult(statistic=0.46214306354522705, pvalue=0.0)
```

After the removal of outliers and test for normality, the categorical data must be addressed prior to model preparation as categorical string values cannot be imputed into the logistic regression classification model. All categorical data was treated with the application of dummy variables which identifies all categorical data into binary indicator variables (Fig. 15). Creating dummy variables is advantageous as it maintains the meaning of the values it encodes in terms of presence or absence of that value, making interpretation of results more direct and fluid. Whereas, if the data was encoded using a label encoder, string values would be assigned a numeric value to represent the string text which doesn't maintain the meaning of values encoded. Also, the label encoder method would add an additional step to interpret results as the numeric value is akin to a placeholder that needs to be translated back to its original value. A disadvantage of using dummy variables is they can lead to an overly-complex model that may overfit and fail to generalize new observations (Taylor, 2022). Given the data's need for encoding, the benefit of dummy variable creation largely outweighs other encoding options necessary to run the logistic regression classifier model.

With all the data cleaned and encoded, the dataframe can be split into training and testing sets where the predictive models will learn from the training set while the test set is used for validation of model efficacy. Using sklearn's train_test_split method, the data was split using an 80/20 ratio where 80% of the data is dedicated to model training while the remaining 20% is used for validation on the test set. Additionally, to ensure an equal split of the target variable, the split was stratified based on deposit_success (Fig. 16). After the data has been split into training and testing sets, the variables must be scaled to give each feature proportional influence on the model as variables use different units of measurements.

Figure 15: One-Hot Encode Categorical Data with get_dummies

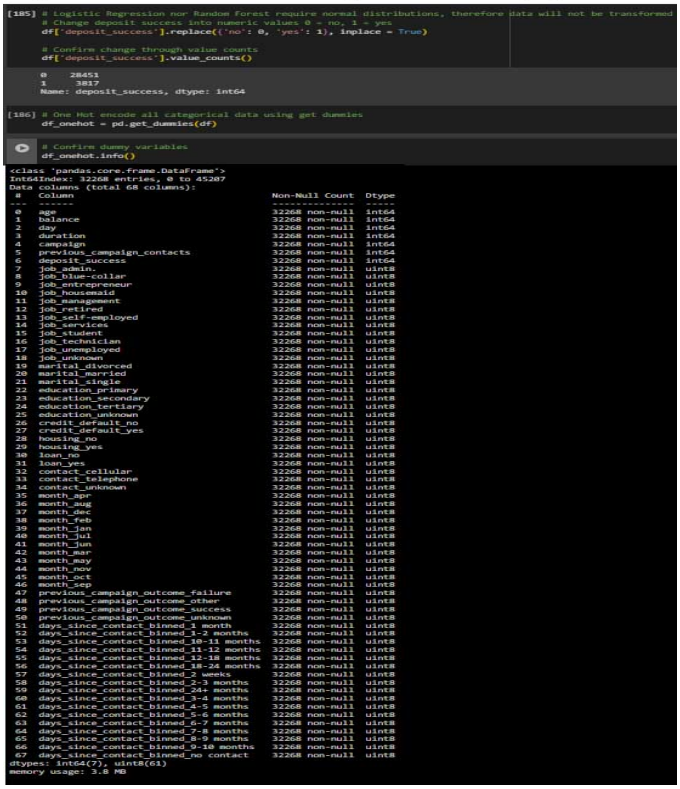



Figure 16: 80/20 Train Test Split with Stratify of Target Feature

```
[ ]  # Split into df_train and df_test as MinMaxScaler need only apply to train set without influence of test set
     y = df_onehot['deposit_success']
     X = df_onehot.drop(columns = ['deposit_success'])

     X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state = 61, stratify = y)

     print("X Train Shape: " + str(X_train.shape))
     print("y Train Shape: " + str(y_train.shape))
     print("X Test Shape: " + str(X_test.shape))
     print("y Test Shape: " + str(y_test.shape))

     X Train Shape: (25814, 67)
     y Train Shape: (25814,)
     X Test Shape: (6454, 67)
     y Test Shape: (6454,)
```

The minmax scaler is employed to set the minimum value of each feature to zero while the maximum value is set to one and the scaler proportionally shrinks the data of each feature within the given range (0 to 1) without changing the shape of the original distribution (Geeks for Geeks, 2022). The fit_transform method is applied to the training data and the minmax scaler simultaneously calculates the range of values within the given range (0 to 1) and scales the values accordingly. To prevent any model bias, the test set is scaled separately using the transform method which scales the test data relative to the parameters learned from fitting the training set (Fig. 17). If the test data is included in the initial fit with the training data, the scaler would be biased as the scaler learned from the data that is to be tested as unseen data. The advantage of using the minmax scaler allows for each feature to maintain their original distribution with equal weight relative to each other, thus, features with larger values don't

negatively influence the model. One disadvantage of using the minmax scaler is if outliers are included in the transformation, the outlying values are preserved and can distort the model. The decision to scale the data is expected to improve model performance and accuracy, but the imbalance of the target variable remains to be treated and must be completed before running the classification models.

Figure 17: MinMax Scaler

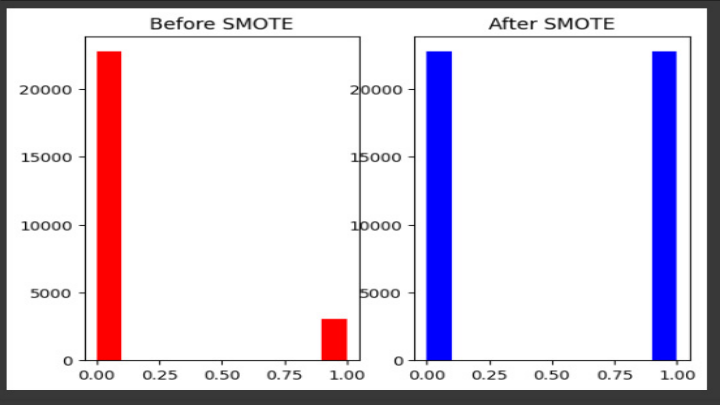
```
[ ]  # Import and instantiate MinMaxScaler
     from sklearn.preprocessing import MinMaxScaler

     # Instantiate MinMaxScaler
     scaler = MinMaxScaler()

     X_train_scaled = scaler.fit_transform(X_train)

     X_test_scaled = scaler.transform(X_test)
```

The final preparations are made by addressing the imbalance of the target feature by applying the synthetic minority oversampling technique (SMOTE). SMOTE is designed to tackle imbalanced datasets by generating synthetic samples to represent the minority class to mitigate bias and capture important features of the minority class, which contribute to a more accurate prediction and improves model performance (Satpathy, 2023). SMOTE was applied to both the minmax scaled X_train and y_train datasets and the results of oversampling the target variable compared to the original imbalance can be visualized in Figure 18. The advantage of using SMOTE allows for the model to learn from additional (synthetic) data that balances the under-represented minority class yielding better performance. However, a disadvantage of SMOTE is there is no consideration for the quality or relevance of the generated synthetic samples, and it is possible that the synthetic samples may not accurately represent the underlying distribution of the minority class (Sole, 2023). The benefits of applying SMOTE largely outweigh any drawbacks and is expected to produce better results compared to imputing imbalanced data into the machine learning classifiers. The data extraction and preparation process is now complete and the data is ready to be analyzed by the logistic regression and random forest classification models.

Figure 18: SMOTE Before/After

```
[74] # Due to imbalance of classification, use SMOTE
     from imblearn.over_sampling import SMOTE

     # Instantiate SMOTE
     smote = SMOTE(random_state = 61)

     # Apply SMOTE to training dataset
     X_train_smote, y_train_smote = smote.fit_resample(X_train_scaled, y_train)
```

```
# Visualize imbalance before and after SMOTE
fig, ax = plt.subplots(1, 2)
ax[0].hist(y_train, color = 'red')
ax[1].hist(y_train_smote, color = 'blue')
ax[0].set_title("Before SMOTE")
ax[1].set_title("After SMOTE")
plt.show()
```




## Part IV: Analysis

**D. Report on the data-analysis process; describe the analysis techniques used, include calculations performed, and their outputs. Justify the selected analysis techniques used, including one advantage and one disadvantage of these techniques.**

The research question aims to identify an accurate predictive model for term deposit success based on a telemarketing campaign and is analyzed using two methods of classification; logistic regression and random forest. Both techniques are classification algorithms that have their strengths and weaknesses of which will be discussed in detail below. The performance of each model is assessed according to their F1 scores, the collective mean between precision and recall. Precision is the ratio of true positive classifications among all the positive cases and recall measures of the models' ability to correctly identify true positives. Effective model performance is gauged in the recognition of positive cases while minimizing false positives and false negatives which is important when dealing with an imbalanced dataset (Buhl, 2023). Both logistic regression and random forest classifiers will learn from the training data and classification reports will be used to assess each model's performance.

Logistic Regression Classifier

Logistic regression is a statistical method used for machine learning models where the dependent variable is dichotomous (or binary) and is used to describe the relationship between the dependent variable and one or more independent variable(s) (Banoula, 2023). The logistic regression algorithm learns from the training data and establishes the best-fitting relationship between deposit success and the independent variables. The best-fit line in logistic regression classification is sigmoid, or S-shaped, and maps predicted values of computed probability values between 0 and 1 (Pant, 2019). Logistic regression classification is advantageous for this analysis as the target variable is binary(yes/no deposit_success) and can establish relationships between the dependent variable and independent variables. Additionally, through model training, the algorithm computes both probability scores for each variable and provides coefficients of feature importance. It is also important to address the disadvantages of logistic regression. These include, independent variables must be linearly related to the log odds and the method is sensitive to extreme outliers; if values differ from the anticipated range erroneous results may occur (Premanand, 2023).

For data analysis, the optimal logistic regression model parameters are explored using the model selection tool, GridSearchCV. Sklearn's logistic regression classifier has numerous options for the solver, therefore, GridSearchCV was employed to identify the best estimator relative to the F1 score (Fig. 19). The parameter optimization utilizes stratified k-fold cross validation and is applied to the scaled and imbalanced dataset because the stratified folds preserve proportional representation of both classes and also prevents influence of synthetic samples from SMOTE. The results of the hyperparameter tuning returned the 'liblinear' solver, which uses a coordinate descent algorithm to solve optimization problems by successively performing approximate minimization along coordinate directions, enforces L1 regularization, and is recommended for large-scale data with high dimensionality (Arnav, 2022). After the logistic regression classifier is instantiated with the optimal parameters, it is fit to the SMOTE training data and results are analyzed using a classification report (Fig. 20). The F1 scores for negative classification returned a value of 0.90 while positive classification returned an F1 score of 0.54. Both scores exceed the threshold set forth by the research question which aimed to identify a predictive model that could exceed an F1 score of 0.30, thus, the null hypothesis can be rejected and the alternate hypothesis accepted. The implications of the model results are further explored through identification of feature importances and model coefficients of which will be discussed in the next section.

Figure 19: Hyperparameter Tuning of Logistic Regression Classifier

```
[71] # Imoport logistic regression
     from sklearn.linear_model import LogisticRegression

     # Instantiate lr for model
     lr = LogisticRegression(random_state = 61, n_jobs = -1)
```

```
[156] # GridSearchCV for Logistic Regression
      from sklearn.model_selection import GridSearchCV
      from sklearn.model_selection import StratifiedKFold

      # Define parameters, look for best solver
      lr_params = {'solver': ['liblinear', 'newton-cg', 'lbfgs', 'sag', 'saga']}

      # Cross validation using StratifiedKFold
      kf = StratifiedKFold(n_splits = 5, shuffle = False)

      # Employ logistic regression GridSearchCV using stratified k-fold cross validation and f1 scoring
      lr_cv = GridSearchCV(lr, lr_params, cv = kf, n_jobs = -1, scoring = 'f1')
```

```
[157] # Fit model to scaled data, not SMOTE data
      lr_grid_fit = lr_cv.fit(X_train_scaled, y_train)

      # Print results
      print("Best LR Parameters: " + str(lr_grid_fit.best_params_))
      print("Best LR Score: " + str(lr_grid_fit.best_score_))

      Best LR Parameters: {'solver': 'liblinear'}
      Best LR Score: 0.45788023564701374
```

```
[204] # Define best regressor based on GridSearchCV results

      # Identify the best estimatora
      lr_best = lr_grid_fit.best_estimator_

      lr_best

      LogisticRegression
      LogisticRegression(n_jobs=-1, random_state=61, solver='liblinear')
```

Figure 20: Logistic Classifier Fit and Report

```
# Fit model
smote_model = lr_best.fit(X_train_smote, y_train_smote)

# Make predictions
pred_lr_smote = smote_model.predict(X_test_scaled)
```

```
[160] # Import classification report
      from sklearn.metrics import classification_report

      # Logistic regressino classification report
      class_report_lr = classification_report(y_test, pred_lr_smote)

      print(class_report_lr)
```

```
              precision    recall  f1-score   support

           0       0.97      0.85      0.90      5691
           1       0.41      0.80      0.54       763

    accuracy                           0.84      6454
   macro avg       0.69      0.82      0.72      6454
weighted avg       0.90      0.84      0.86      6454
```

Random Forest Classifier

The random forests classifier is a machine learning technique composed of numerous decision trees that operate as an ensemble whereby each individual tree in the random forest results with a class prediction and the class with the most votes (classification) becomes the model's prediction (Yiu, 2019). The building blocks of a random forest, the decision tree, is composed of a root node, internal node, branches, and leaf nodes. The root node is the inception of the classification task, a starting point with no previous branches. The outgoing branches from the root node then feed into the internal nodes (or decision nodes) and based on the available features, both node types conduct evaluations to form homogenous subsets, which are denoted by leaf nodes that represent all the possible outcomes within the dataset (IBM, n.d.). To avoid correlation between decision trees within the random forest, two methods assist in maintaining model integrity, bootstrap aggregation (bagging) and feature randomness. Bagging is the process where each individual tree takes a random sample from the data with replacement (resulting in different trees) while enforcing feature randomness at decision nodes guarantees variation amongst the trees within the model, ultimately lowering correlation between trees and improves diversification (Yiu, 2019). Implementing a random forest classifier for term deposit success is advantageous because the machine learning algorithm can be used in classification analysis, isn't sensitive to outliers, outputs feature importances, and the model won't overfit the data when there are enough trees in the forest (Donges & Urwin, 2023). The disadvantage of a random forest classifier is the computational power demand required for accurate predictions because more decision trees are required to improve accuracy which comes at a tradeoff with increased run-time and a slower performing model (Great Learning Team, 2023).

Similar to the logistic regression model, the random forest classifier undergoes hyperparameter tuning through the model selection tool GridSearchCV and the best estimator is evaluated relative to the F1 score (Fig. 21). As with the logistic classifier, the parameter optimization implements a stratified k-fold validation and is applied to the scaled, imbalanced data. This preserves equal representation of the binary classification, excluding any influence of synthetic samples from SMOTE. Following the hyperparameter tuning, the random forest classifier is instantiated with the best estimator, then is fit to the SMOTE training data, and analyzed using a classification report (Fig. 22). The F1 scores for negative classification returned a value of 0.91 while positive classification returned an F1 score of 0.54. Comparable to the logistic regression classifier, the results of these F1 scores exceed the threshold proposed by the research question. The null hypothesis can be rejected while confirming the alternate hypothesis that a random forest predictive model can achieve an F1 score that exceeds 0.3. The implications of the model results are further explored through identification of feature importances and will be discussed with the logistic regression classifier results in the next section.

Figure 21: Hyperparameter Tuning of Random Forest Classifier

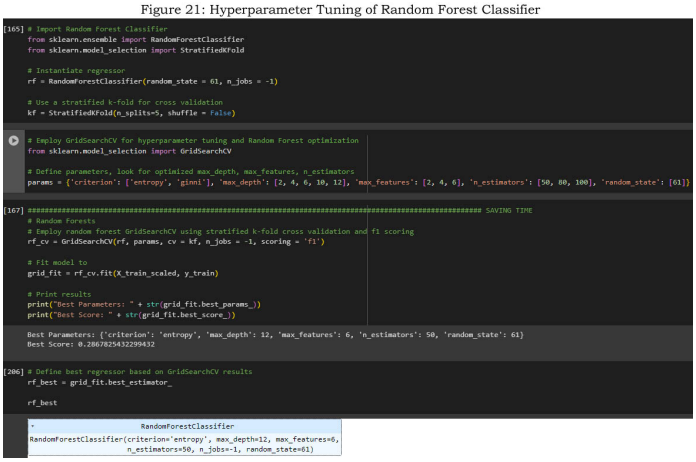
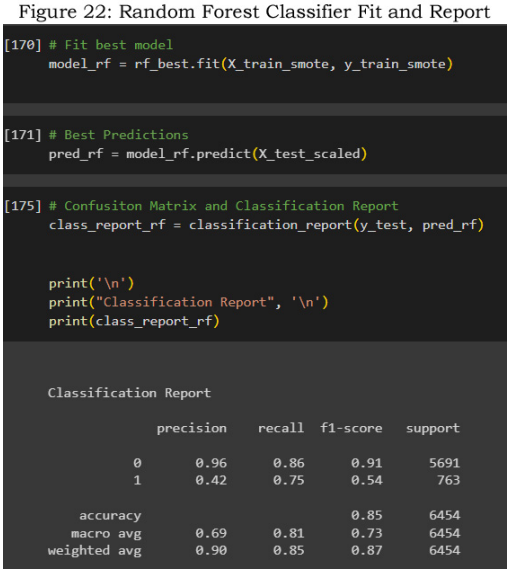



Figure 22: Random Forest Classifier Fit and Report




# Part IV: Data Summary and Implications

**E. Summarize the implications of the data analysis and discuss the results in context of the research question. Include one limitation of your analysis, recommend a course of action based on the results, and propose two directions for future study of the data set.**

The research project analyzed data from a bank that collected 45,211 records from a telemarketing campaign which contained 16 explanatory variables and the target variable, deposit_success. To ensure optimal model performance, these features were explored, cleaned, and engineered before being fit by logistic regression and random forest machine learning classifiers. The data had no missing values and contained zero duplicate records, however, through data exploration, statistical outliers were found and treated using the interquartile method. A total of 12,940 outliers were detected and removed from the model data and isolated into a separate “outlier” dataframe (note: three additional records were removed due to zero

values for duration). The remaining 32,268 records were split into training and testing sets where 80% of the data was used for training while 20% was withheld for testing model efficacy. The data was scaled using a minmax scaler and SMOTE (synthetic minority over-sampling technique) was applied to better represent the minority class. For optimal model performance, hyperparameter tuning was implemented to find the best metrics for both logistic regression and random forest classifiers. The best estimators of each machine learning model were fit to the training data and model performances were measured relative to F1 scores calculated in classification reports. The research question set out to discover if term deposits could be predicted using logistic regression and random forest classifiers from imbalanced data with an F1 score greater than 0.3.

Logistic Regression Analysis of Results

The results from the logistic regression classifier returned F1 scores of 0.85 for predicting “no” to term deposits and 0.54 for predicting “yes” to term deposits. Both of these values exceed the threshold proposed by the research question and confirm a logistic regression classifier is capable of predicting term deposit success from imbalanced data with an F1 score above 0.3. As discussed previously, the F1 score is an evaluation metric that measures a model’s accuracy by computing the harmonic mean of precision and recall scores; where precision measures the ratio of how many of the “positive” predictions were correctly labeled positive (false positive classifications decreases precision), while recall evaluates how many of the positive class samples present in the dataset were correctly identified as positive (Kundu, 2022). The model’s ability to predict negative classifications was inherently expected due to the imbalance of data because without a model it was known that approximately 88% of all customers elected not to commit to a term deposit with the banking institution. The 0.90 F1 score for negative classifications averaged a precision value of 0.97 and recall score of 0.85, thus, the model’s ability to identify customers who will not commit to term deposits is highly accurate. The 0.54 F1 score for positive classifications averaged a positive precision score of 0.41 and recall value of 0.80, exceeding the expectation posed by the research question regarding the model’s ability to accurately identify term deposit success. The 0.8 recall score for positive term deposit classification informs the model’s ability to accurately identify the positive cases but comes with a tradeoff of a lower precision of 0.41. This means, the model is also identifying a large number of false positive cases which reduces the overall precision score. Given the model’s performance, it is important to delve deeper into the results and glean further understanding of feature influences.

One of the advantages of logistic regression classification is the ability to extract feature coefficients which inform the impact of that feature’s presence on predictive classifications. The coefficients output by the logistic model indicate the change in the expected log odds relative to a one unit change in a variable, holding all other predictors constant (Boston University School of Public Health, 2013). The coefficients of the logistic classification model were put into a new dataframe, sorted in decreasing order of their absolute values, and visualized on a bar plot (Fig. 23). When viewing the top-10 of the feature importances, there is a

mix of positively and negatively influential variables, with the most impactful variable being duration (Fig. 24). Therefore, it is apparent that the amount of time spent speaking with the customer had an overall positive influence in the model's predictions. The logistic classification model provides a strong foundation in predictive ability, but can be further improved upon as limiting factors need to be taken into account when assessing the results.

Figure 23: Bar Plot of Feature Importances from Logistic Regression Classification Model

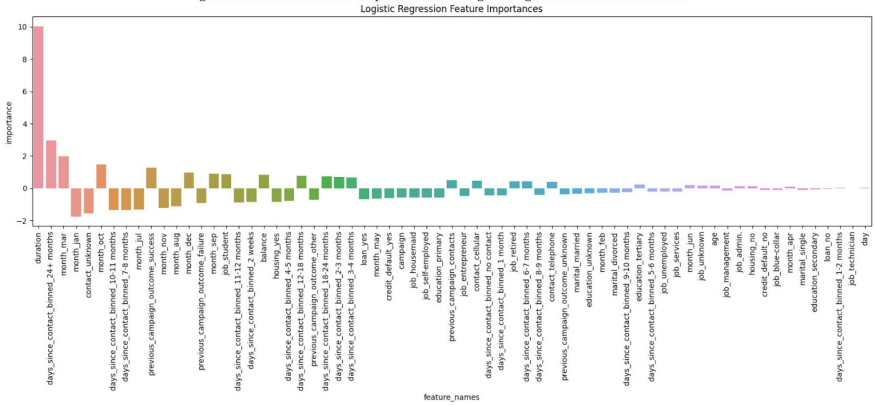



Figure 24: Top 10 Logistic Regression Feature Importances

```
# Sort features relative to absolute value of importances
lr_sort = lr_sort.sort_values(by = 'abs_importance', ascending = False)

# View the top 10
lr_sort[['feature_names', 'importance']].head(10)
```

| | feature_names | importance |
|---|---|---|
| 3 | duration | 10.034 |
| 58 | days_since_contact_binned_24+ months | 2.959 |
| 41 | month_mar | 1.967 |
| 38 | month_jan | -1.753 |
| 33 | contact_unknown | -1.564 |
| 44 | month_oct | 1.460 |
| 52 | days_since_contact_binned_10-11 months | -1.366 |
| 63 | days_since_contact_binned_7-8 months | -1.358 |
| 39 | month_jul | -1.321 |
| 48 | previous_campaign_outcome_success | 1.285 |

## Random Forest Analysis of Results

The results from the random forest classifier output F1 scores of 0.91 for predicting "no" to term deposits while predictions of "yes" to term deposits returned an F1 score of 0.54. The relevance of the scores compared to the proposed research question confirms that a random forest classification model can accurately predict term deposit success from imbalanced data with an F1 score that exceeds 0.3. As mentioned in the analysis of the logistic regression classifier, the ability to project negative classifications was innately expected due to the imbalance of data, therefore, further study of the positive and negative predictions provide deeper understanding of model performance. The 0.91 F1 score for the random forest model's negative classifications averaged a precision value of 0.96 and recall score of 0.86, thus, the model's ability to identify

customers who will not complete term deposits is slightly better than the predictive power of the logistic model (increase of 0.01). Whereas, the 0.54 F1 score for positive classifications averaged the precision score of 0.42 and recall of 0.75, a nominal increase in precision and slight decrease in recall when compared to the logistic model. Overall, the F1 score for the random forest classifier remained identical to that of the logistic classifier as both models surpassed the F1 score proposed by the research question. Similar to the logistic model, the 0.75 recall score for positive term deposit classification identifies the model's strong ability to accurately identify the positive cases. However, there is a slight tradeoff with a lower precision score of 0.42, as the random forest model identifies a higher number of false positive cases which decreases the overall precision score. The performance of the random forest classification model has been proven successful in regards to the research question, but understanding the role of features' influence on the model is additionally paramount.

Comparable to the logistic regression classifier model, feature importances can also be extracted from the random forest model, however, there are some drawbacks. The greatest drawback is the feature importances from the random forest model are all positive values, meaning, the values don't provide any information about the direction of the relationship between the feature and the target (Filho, 2023). The feature importance values of the random forest model were placed into a new dataframe, sorted in decreasing order, and viewed on a bar plot (Fig. 25). When observing the top-10 importances, duration was found to be the most influential feature in the random forest classifier (Fig. 26). Both models have identified duration as the most prominent feature in determining term deposit classifications; reinforcing the importance of effective communication with customers. While both models performed well, it is also imperative to be cognizant of the study's limitations when evaluating the results and proposing courses of action.

Figure 25: Bar Plot of Feature Importances from Random Forest Classification Model

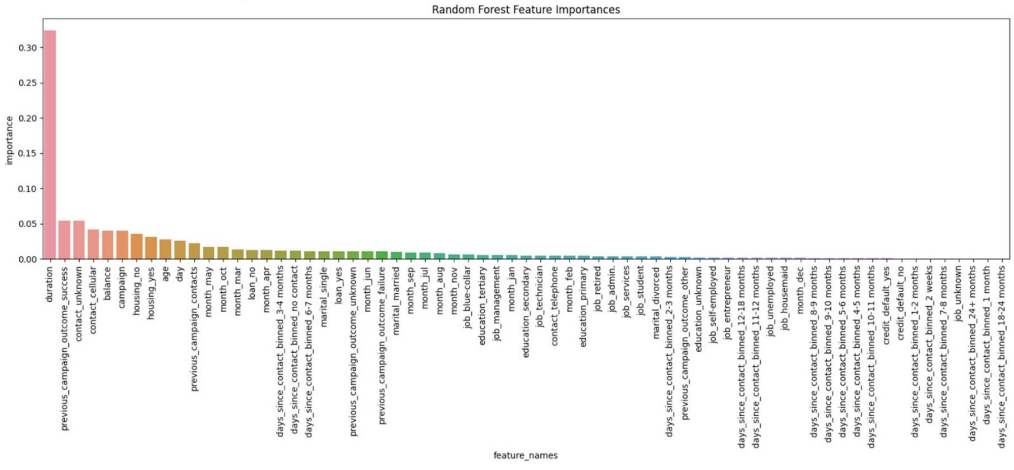
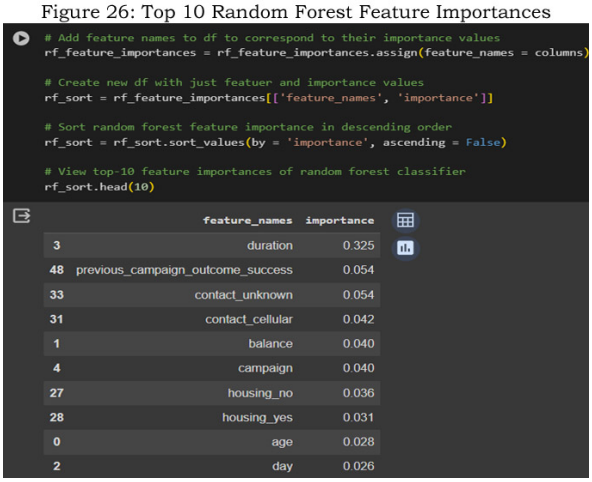


Figure 26: Top 10 Random Forest Feature Importances



```
# Add feature names to df to correspond to their importance values
rf_feature_importances = rf_feature_importances.assign(feature_names = columns)

# Create new df with just featuer and importance values
rf_sort = rf_feature_importances[['feature_names', 'importance']]

# Sort random forest feature importance in descending order
rf_sort = rf_sort.sort_values(by = 'importance', ascending = False)

# View top-10 feature importances of random forest classifier
rf_sort.head(10)
```

| | feature_names | importance |
|---|---|---|
| 3 | duration | 0.325 |
| 48 | previous_campaign_outcome_success | 0.054 |
| 33 | contact_unknown | 0.054 |
| 31 | contact_cellular | 0.042 |
| 1 | balance | 0.040 |
| 4 | campaign | 0.040 |
| 27 | housing_no | 0.036 |
| 28 | housing_yes | 0.031 |
| 0 | age | 0.028 |
| 2 | day | 0.026 |

## Limitation

The goal of the analysis aimed to identify the construction of machine learning models capable of predicting term deposit success from imbalanced data with an F1 score greater than 0.3 using logistic regression and random forest classifiers. Overall, both logistic regression and random forest classifiers successfully answered the research question, confirming that term deposits can be accurately predicted as both models returned similar F1 scores for both positive and negative classifications (all exceeding 0.3 threshold posed by the research question). However, it is important to understand there are limitations to the conclusions drawn from this analysis. One such limitation in this study is the removal of outliers previous to fitting the models. The dataset started with 45,211 records and after data cleaning and removal of outliers, 32,268 records were used for model training and testing. A difference of 12.943, meaning, when outliers were removed, roughly 28.6% of the records were excluded from the analysis. While it is important to include only the most statistically significant records for analysis, removing such a high volume of outliers is a considerable limitation of this study.

## Course of action

Based on results from both predictive models, valuable insight has been gained in projecting term deposit success with F1 scores greater than 0.3. There are three courses of actions that would build upon the foundation this analysis has established. The first, take a deeper look into the identified feature importances from both models. Each model identified duration as the most influential feature, but taking a more in-depth look at the other influential features could help bolster future marketing campaigns. Also, provided duration had a substantial effect on term deposits, refining future marketing campaigns to reflect effective and informative contact in terms of duration may increase future term deposit success. The second course of action would be to take the outliers and employ the same analysis to the outlier dataset. This includes; applying the minmax scaler, train and test split, implementing SMOTE, identifying new

hyperparameters for the classifiers with GridSearchCV, and running new logistic regression and random forest models. Then use the F1 scores to compare results of the outlier models to the original models and look for any similarities or differences as well as examine feature importances of the outlying data. Lastly, the bank could implement the current model(s) from this analysis in predicting term deposit success. Both logistic regression and random forest classifiers returned identical F1 scores (0.54) for positive classification of term deposit success. However, the edge would lean towards the logistic regression classifier due to the higher recall score, which defines the model's ability to accurately identify positive cases among all true positives classifications. Employing the logistic regression classifier to predict future term deposit cases will prove beneficial to the banking institution. While these actions can be utilized immediately, it is also worth considering additional directions for future studies of the dataset.

Two directions for future study of dataset

The initial analysis of the bank dataset has proved fruitful, but understanding the next progression of analysis is equally important. Therefore, two directions for future study should be considered. One, is it possible to improve the analysis through different data treatments and implementing the same machine learning models. Two, are there other machine learning models worth considering for classification of imbalanced data.

First and foremost, it is crucial to know if the F1 metric from this analysis can be improved upon. While the F1 scores for both the logistic regression and random forest classifiers confirmed the alternate hypothesis of the research question, it is important to see if the models can be augmented. This study removed outliers and implemented successful predictive models which returned positive results, but it is essential to know if the models could be enhanced. Therefore, it is suggested that the method for addressing outliers be adjusted in a way that doesn't use the interquartile method. Rather, remove extreme outliers based on standard deviations relative to the mean or median (depending on data distribution) and retain 95% of a feature's distribution. For data cleaning with this method, variables that have a normal distribution are treated by retaining all data within two standard deviations from the mean. Similar to the upper and lower limits of the interquartile method, the upper boundary is plus two standard deviations from the mean while the lower boundary is minus two standard deviations from the mean and outliers are all values that reside beyond the upper and lower boundaries. If the data distribution of a variable is skewed, the values retained should be plus/minus two standard deviations from the median. Given the removal of extreme outliers and utilizing the minmax scaler, significantly less outliers will be removed from analysis and could bolster term deposit predictive power as well as identify other meaningful feature importances.

Second, it is worth considering the possibility that other machine learning algorithms could outperform the logistic regression classifier and/or the random forest classifier. While this analysis achieved accurate models, there are more powerful algorithms capable of binary classifications and are not sensitive to outliers. Therefore, it is suggested that machine learning models not sensitive to outliers be considered in further analysis. For example, examine model

performance using XGBoost, AdaBoost, and/or Naive Bayes algorithms. These machine learning methods are robust to outliers, thus, data treatment and removal can be minimal, while maximizing learning model input. Furthermore, feature importances can be extracted from each method and compared to those identified by the logistic regression and random forest classifiers.

In terms of the research question, this analysis has been successful in identifying two machine learning models capable of predicting successful term deposits with F1 scores that exceed 0.3. Both logistic regression and random forest classifiers performed well and both achieved similar F1 scores. However, it is imperative to be thorough in one's approach to data analysis, thus, it is essential to consider alternative data treatment methods and machine learning models. One such approach would be to treat outlier removal differently and include 95% of data distributions previous to scaling the data and model training/testing. Lastly, the consideration of other machine learning algorithms not sensitive to outliers may lead to better results and conclusions. Through the addition of further downstream testing, one can be confident with the results after exhausting data treatment and modeling options to complete a thorough analysis.

## F. Sources

### LITERARY SOURCES

Arnav, R. (2022, March 22). Scikit-learn solvers explained. Medium. https://medium.com/@arnavr/scikit-learn-solvers-explained-780a17bc322d#.

Arya, N. (2022, July 25). Does the Random Forest Algorithm Need Normalization?. KDNuggets. https://www.kdnuggets.com/2022/07/random-forest-algorithm-need-normalization.html.

Banoula, M. (2023, November 7). An Introduction to Logistic Regression in Python. SimpliLearn. https://www.simplilearn.com/tutorials/machine-learning-tutorial/logistic-regression-in-python.

Boston University School of Public Health. (2013, January 17). Multiple Logistic Regression Analysis. Boston University School of Public Health. https://sphweb.bumc.bu.edu/otlt/mph-modules/bs/bs704_multivariable/bs704_multivariable8.html.

Buhl, N. (2023, July 18). F1 Score in Machine Learning. Encord. https://encord.com/blog/f1-score-in-machine-learning/.

Donges, N. & Urwin, M. (2023, September 15). Random Forest: A Complete Guide for Machine Learning. Builtin. https://builtin.com/data-science/random-forest-algorithm#procon.

Filho, M. (2023, March 30). How To Get Feature Importance in Random Forests. Forcastegy. https://forecastegy.com/posts/feature-importance-in-random-forests/.

Frost, J. (2023, May 18). Interquartile Range (IQR): How to Find and Use It. Statistics by Jim. https://statisticsbyjim.com/basics/interquartile-range/.

Geeks for Geeks. (2022, February 3). Data Pre-Processing with Sklearn using Standard and Minmax Scaler. Geeks for Geeks. https://www.geeksforgeeks.org/data-pre-processing-wit-sklearn-using-standard-and-minmax-scaler/.

Geeks for Geeks (n.d.) Random Forest Classifier using Scikit-learn. Geeks for Geeks. Retrieved December 31, 2023. https://www.geeksforgeeks.org/random-forest-regression-in-python/.

Great Learning Team. (2023, June 13). Random forest Algorithm in Machine learning: An Overview. MyGreatLearning. https://www.mygreatlearning.com/blog/random-forest-algorithm/.

IBM. (n.d.). What is a Decision Tree?. IBM. Retrieved December 30, 2023. https://www.ibm.com/topics/decision-trees#.

Jiang, E., Wang, Z., & Zhao, J. (2022, December 12). Prediction of Term Deposit in Bank: Using Logistic Model. BCP Business & Management. http://dx.doi.org/10.54691/bcpbm.v34i.3071.\

Kumar, S. (2021, August 25). Essential guide to handle Outliers for your Logistic Regression Model. Medium. https://medium.com/geekculture/essential-guide-to-handle-outliers-for-your-logistic-regression-model-63c97690a84d.

Kundu, R. (2022, December 16). F1 Score in Machine Learning: Intro & Calculation. V7 Labs. https://www.v7labs.com/blog/f1-score-guide.

Moro, S., Cortez, P., & Rita, P. (2014, June 1) A Data-Driven Approach to Predict the Success of Bank Telemarketing. Decision Support Systems, Elsevier. https://doi.org/10.1016/j.dss.2014.03.001.

Muslim, MA., Dasril, Y., Alamsyah, A., & Mustaqim, T. (2021, June 1). Bank predictions for prospective long-term deposit investors using machine learning LightGBM and SMOTE. Journal of Physics. https://iopscience.iop.org/article/10.1088/1742-6596/1918/4/042143.

Pant, A. (2019, January 22). Introduction to Logistic Regression. Towards Data Science. https://towardsdatascience.com/introduction-to-logistic-regression-66248243c148.

Premanand, S. (2023, April 27). Building an End-to-End Logistic Regression Model. Analytics Vidhya. https://www.analyticsvidhya.com/blog/2021/10/building-an-end-to-end-logistic-regression-model/
.
Raj, A. (2020, November 7). Perfect Recipe for Classification Using Logistic Regression. Towards Data Science. https://towardsdatascience.com/the-perfect-recipe-for-classification-using-logistic-regression-f8648e267592.

Roy, K., Kar, S., & Das, R.N. (2015, March 3). Selected Statistical Methods in QSAR. Understanding the Basics of QSAR for Applications in Pharmaceutical Sciences and Risk Assessment. https://doi.org/10.1016/B978-0-12-801505-6.00006-5.

Satpathy, S. (2023, November 17). SMOTE for Imbalanced Classification with Python. Analytics Vidhya. https://www.analyticsvidhya.com/blog/2020/10/overcoming-class-imbalance-using-smote-techniques/.

Sole. (2023, March 28). Overcoming Class Imbalance with SMOTE: How to Tackle Imbalanced Datasets in Machine Learning. Train in Data. https://www.blog.trainindata.com/overcoming-class-imbalance-with-smote/.

Taylor, M. (2022, January 19). Dummy Variables. Vexpower. https://www.vexpower.com/brief/dummy-variables.

Yiu, T. (2019, June 12). Understanding Random Forest. Towards Data Science. https://towardsdatascience.com/understanding-random-forest-58381e0602d2.

Zach. (2021, May 19). How to Interpret an Odds Ratio Less Than 1. Statology. https://www.statology.org/interpret-odds-ratio-less-than-1/.