ADS500B Data Science Programming

Final Project: Data Analysis and Preliminary Analytics

Stephen Reagin and Trevor Sauerbrey

August 15, 2022

Outline of Group Presentation

GitHub repository: https://github.com/sfreagin/ADS_500B_project

Choosing a Dataset

We were given three datasets to review for our group project. The first step was to review each dataset and the corresponding data dictionaries/README files to understand what we had.

Dataset 1 – Bank Marketing
This dataset consists of 17 variables, of which 7 are numerical / quantitative. Of the available datasets, this one had the most total data—more than 45,000 rows and 17 features, with almost no missing data points. It also has a good mix of quantitative and categorical variables, and the data dictionary was very useful describing both the features and the overall purpose of the dataset.

Dataset 2 – Housing Data
This dataset includes data for houses sold over a 1-year period, from May 2014 to May 2015, and consists of more than 21,000 rows with 21 features. The most obvious open question was to ask, can we predict housing prices based on the given data features? This would most likely be a multivariate regression model; however, we weren't very interested in this question.

Dataset 3 – Online Shoppers
This dataset consists of more than 12,000 rows with 18 features. We rejected this dataset almost immediately for a number of reasons. In no particular order:

- The corresponding data dictionary was rather unhelpful in describing both the details of feature variables, as well as the overall purpose of the full dataset
- The distribution of categorical variables didn't make much sense—for example, more than 5000 records between March and May but 0 records in the month of April
- Numerical variables tended to have long-skew distributions, and there were many imbalanced data classes for categorical variables plus lots of zeroes

Thus, we chose Dataset #1 because it had the most data to work with, a good mix of categorical or qualitative vs. quantitative variables with relatively few missing data points, and a good data dictionary/README description to understand the purpose of this dataset (predicting deposits). We were also able to apply lessons learned from our previous USD class in *Probability and Statistics* on navigating Type I and Type II errors within classification problems.

Exploratory Data Analysis

The Bank Marketing dataset is a CSV file based on the marketing campaigns of a Portuguese banking institution, with observations recorded from May 2008 to November 2010. The purpose of these campaigns was to encourage potential clients to sign up for a bank term deposit, which was successfully recorded for 11.7% of all observations. The dataset has more than 45,000 observations of 17 total variables, 10 of which are naturally categorical and 7 of which are given to us as numerical / integers / continuous values.

Structure of the Dataset

The first step was to quickly understand the available variables within the dataset. Pulling the CSV file into a Pandas (Python) dataframe and taking a quick look at the first several rows.

	age	job	marital	education	default	balance	housing	loan	contact	day	month	duration	campaign	pdays	previous	poutcome	deposit
0	58.0	management	married	tertiary	no	2143	yes	no	unknown	5	may	261	1	-1	0	unknown	no
1	44.0	technician	single	secondary	no	29	yes	no	unknown	5	may	151	1	-1	0	unknown	no
2	33.0	entrepreneur	married	secondary	no	2	yes	yes	unknown	5	may	76	1	-1	0	unknown	no
3	47.0	blue-collar	married	unknown	no	1506	yes	no	unknown	5	may	92	1	-1	0	unknown	no
4	33.0	unknown	single	unknown	no	1	no	no	NaN	5	may	198	1	-1	0	unknown	no

For numerical quantities, we can also take a quick look at the summary statistics—mean, standard deviation, IQR, etc.

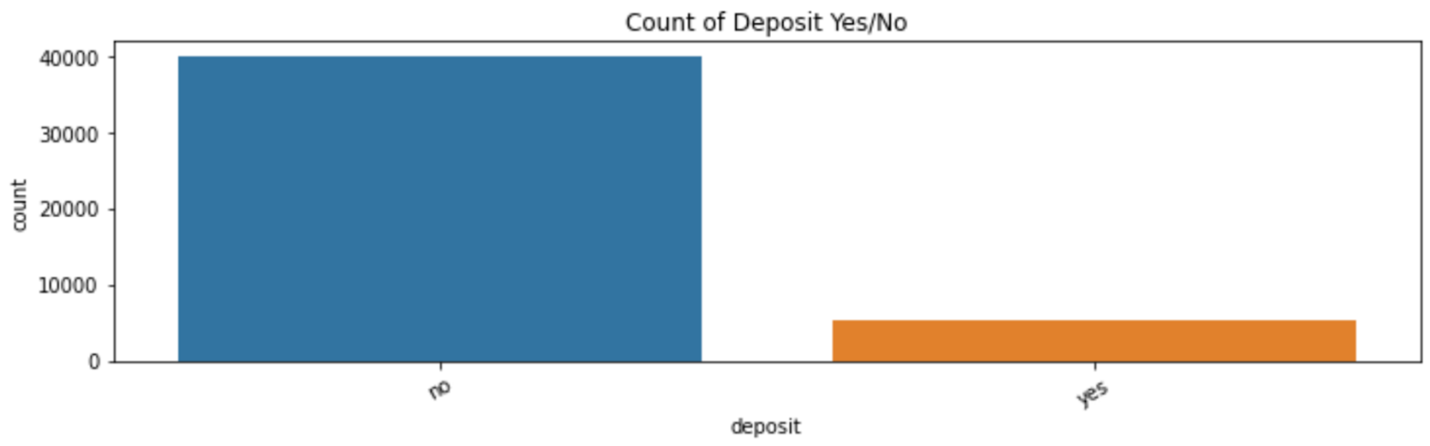
	age	balance	day	duration	campaign	pdays	previous
count	43872.000000	45211.000000	45211.000000	45211.000000	45211.000000	45211.000000	45211.000000
mean	40.924781	1362.272058	15.806419	258.163080	2.763841	40.197828	0.580323
std	10.610835	3044.765829	8.322476	257.527812	3.098021	100.128746	2.303441
min	18.000000	-8019.000000	1.000000	0.000000	1.000000	-1.000000	0.000000
25%	33.000000	72.000000	8.000000	103.000000	1.000000	-1.000000	0.000000
50%	39.000000	448.000000	16.000000	180.000000	2.000000	-1.000000	0.000000
75%	48.000000	1428.000000	21.000000	319.000000	3.000000	-1.000000	0.000000
max	95.000000	102127.000000	31.000000	4918.000000	63.000000	871.000000	275.000000

It should be noted, however, we later converted **day** to a categorical variable, because it refers to *day of the month* that a client was contacted, according to the data dictionary, and should not be treated as a numerical or ordinal variable. We also later replaced **pdays** and **previous** with a binary Yes/No variable, as will be described later.

Deposit

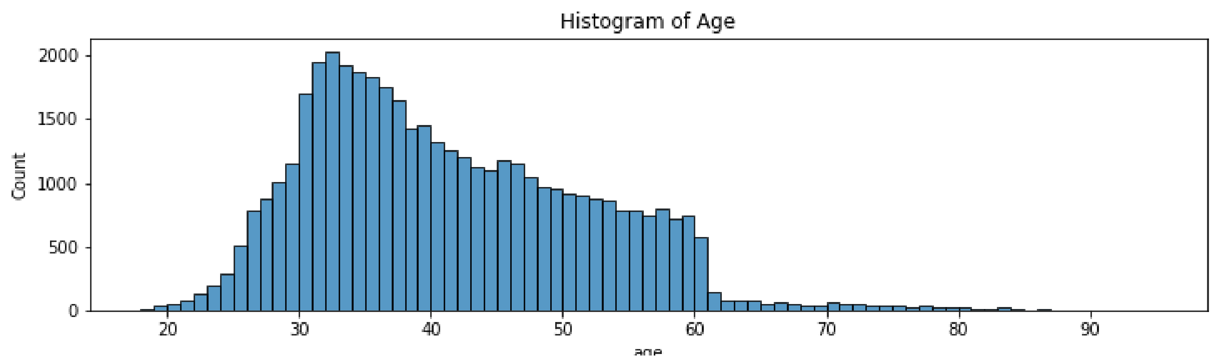
This dataset was originally created to understand the **deposit** variable, to help drive future customer outcomes towards a "yes" decision. The dataset itself is imbalanced.

```
no      0.883015
yes     0.116985
Name: deposit, dtype: float64
```

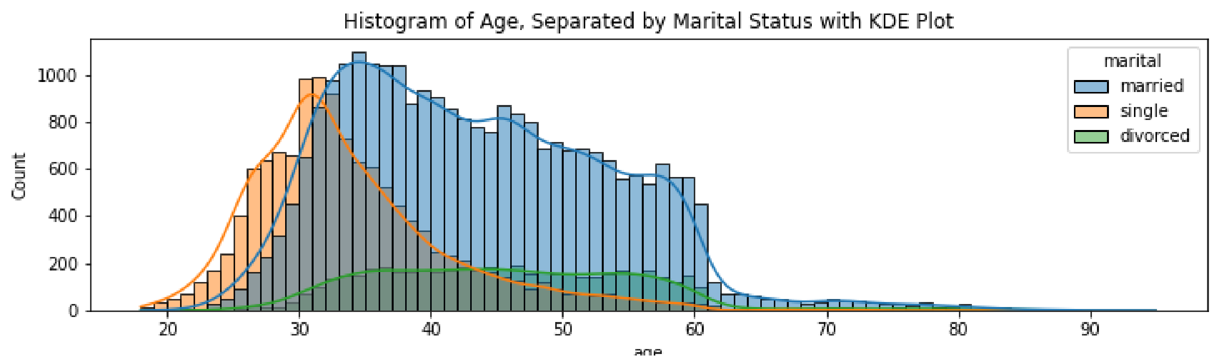


Age

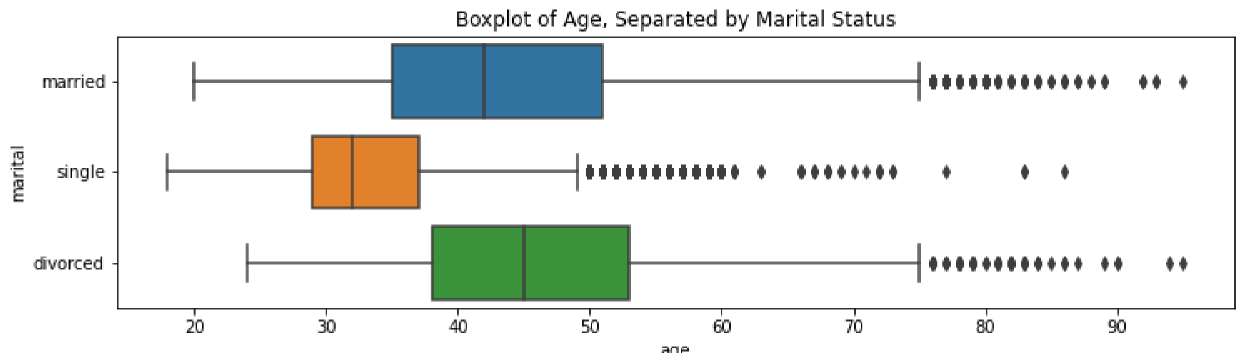
We plotted histograms for the **age** variable, and the distribution at least "makes sense" in that there seems to be a continuous probability distribution:



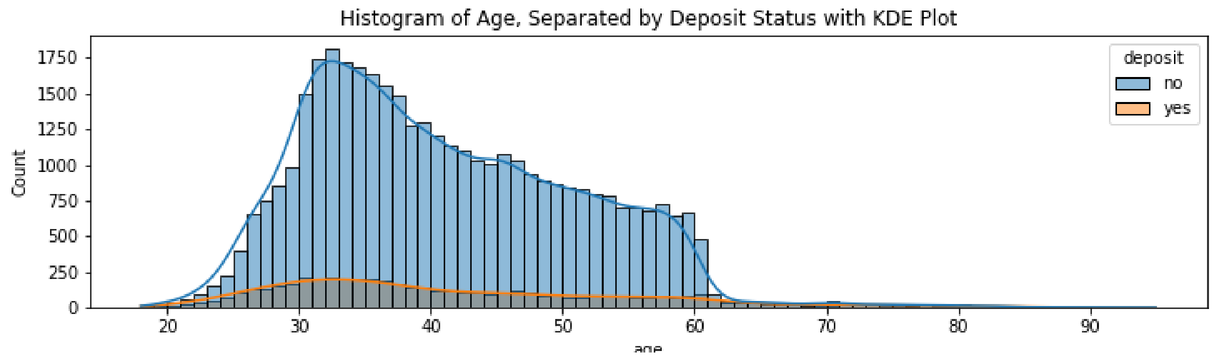
As another sanity check, we created another histogram and boxplot separated by **marital** status.

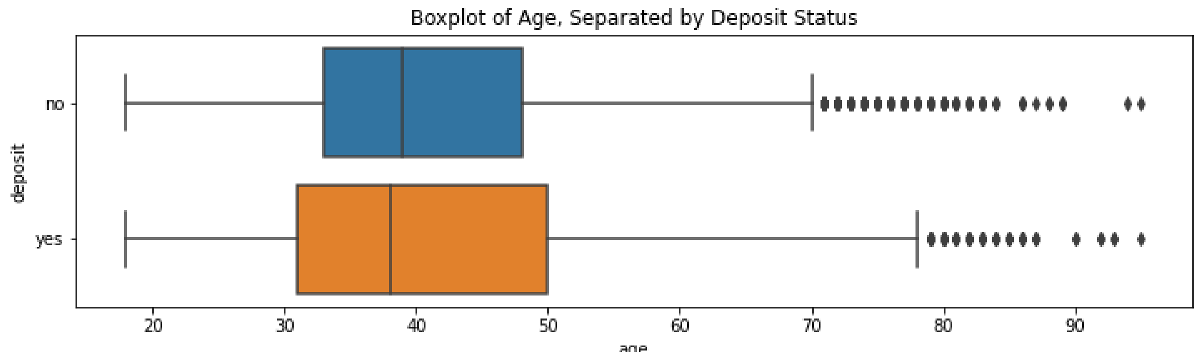


The findings align with common sense understanding, namely that single people tend to be younger, followed by married people, and lastly divorced people tend to be older—after all, you cannot get divorced before you get married.



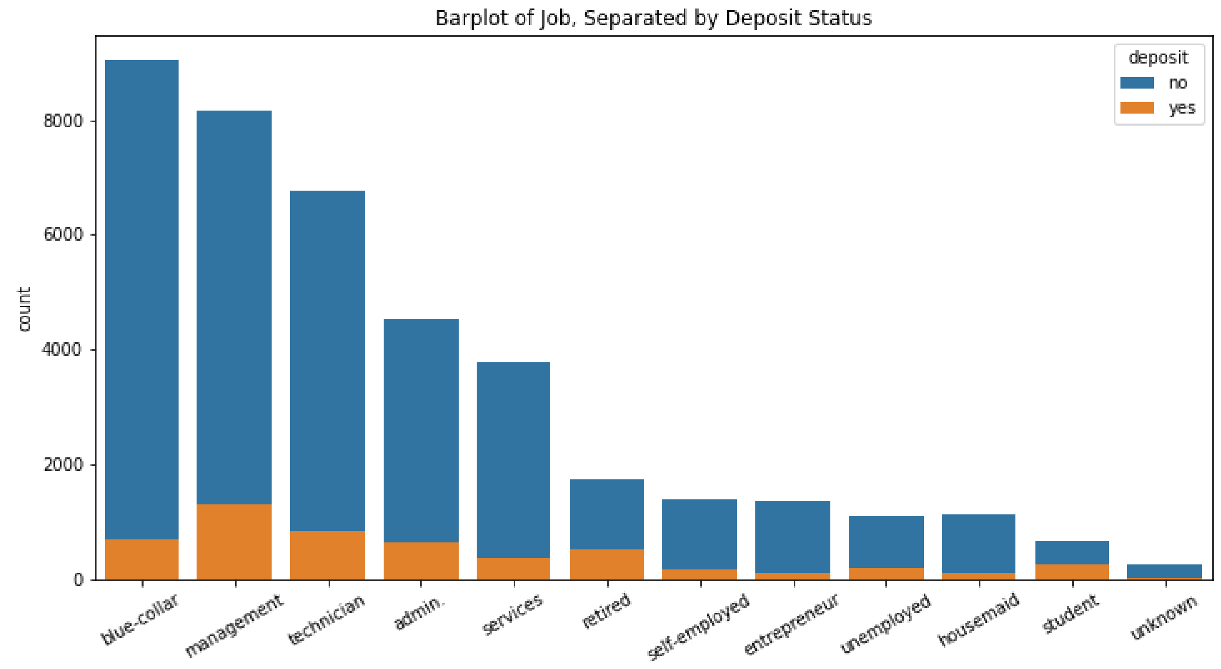
However, the distribution for **age** does not seem to change much when separated by **deposit** status. That is to say, age does not seem to be a reliable indicator for whether a person is more or less likely to sign up for a bank term deposit.





Job

When reviewing the **job** variable, there seems to be some variation in proportions of Yes/No for **deposit**, although the sample sizes do get rather small and thus may be less reliable indicators for certain **job** categories.

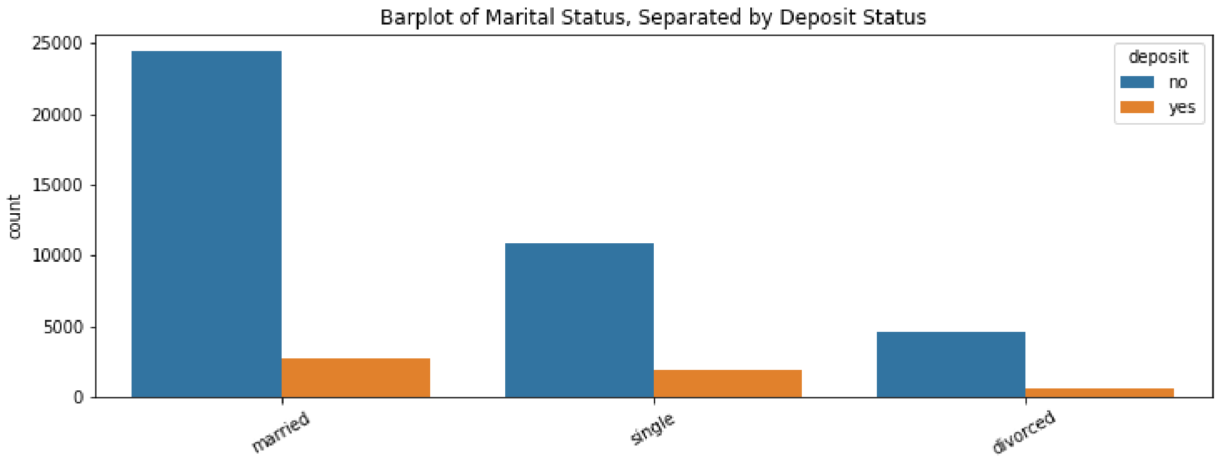


Percentages:

job	admin.		blue-collar		entrepreneur		housemaid		management		retired		self-employed		services		student		technician		unemployed		unknown	
deposit	no	yes	no	yes	no	yes	no	yes	no	yes	no	yes	no	yes	no	yes	no	yes	no	yes	no	yes	no	yes
deposit	0.88	0.12	0.93	0.07	0.92	0.08	0.91	0.09	0.86	0.14	0.77	0.23	0.88	0.12	0.91	0.09	0.71	0.29	0.89	0.11	0.84	0.16	0.88	0.12

Marital

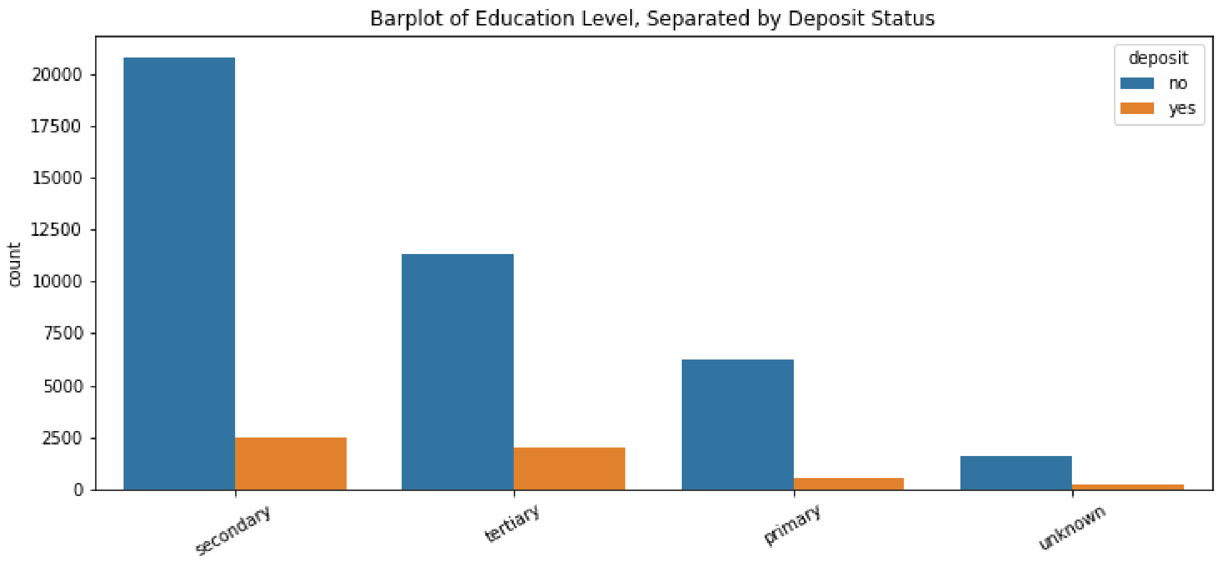
A change in **marital** status doesn't seem to have a dramatic change in **deposit** status.



marital	divorced		married		single	
deposit	no	yes	no	yes	no	yes
deposit	0.88	0.12	0.9	0.1	0.85	0.15

<u>Education</u>

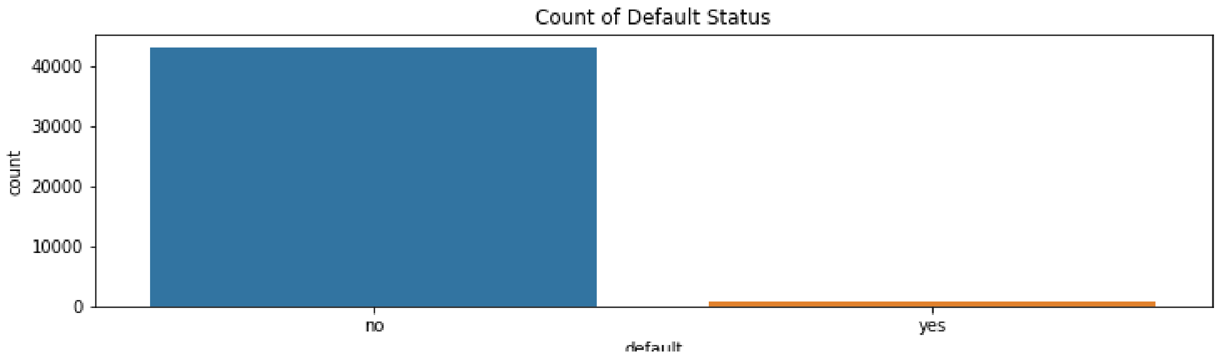
A change in **education** level doesn't seem to have a dramatic effect on **deposit** status.



<u>Default</u>

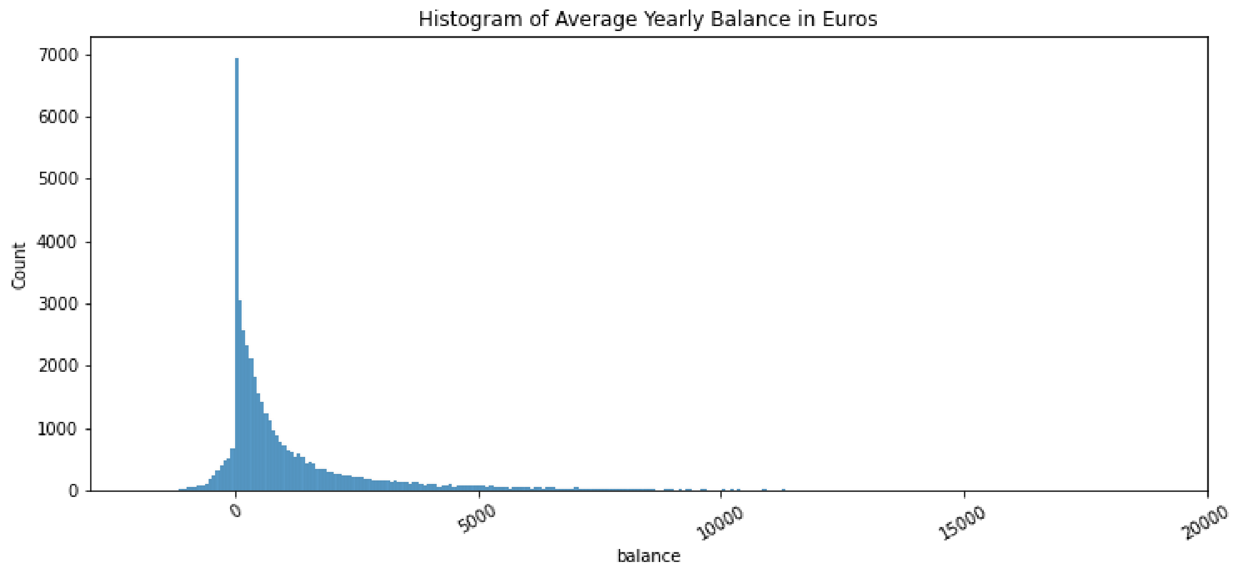
This refers to whether or not a potential client has credit in default, and is marked by a binary Yes/No. While there does seem to be a difference in **deposit** status rate based on **default** status, the overall **default** classes are so imbalanced that this may be a small sample size difference.

default	**no**		**yes**	
deposit	**no**	**yes**	**no**	**yes**
deposit	0.88	0.12	0.93	0.07

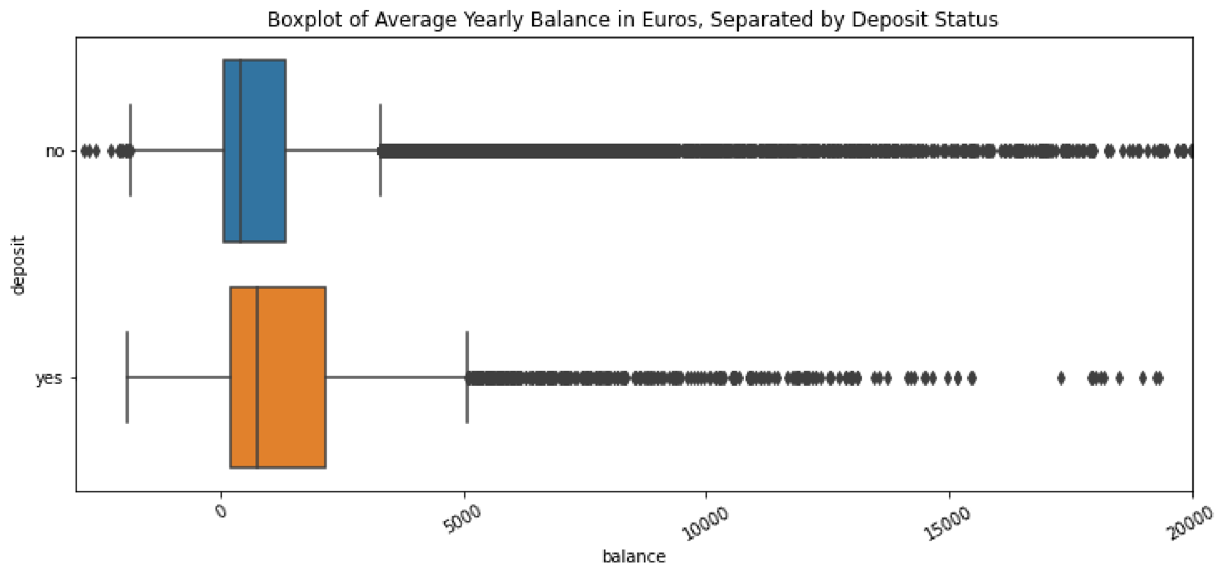


Balance

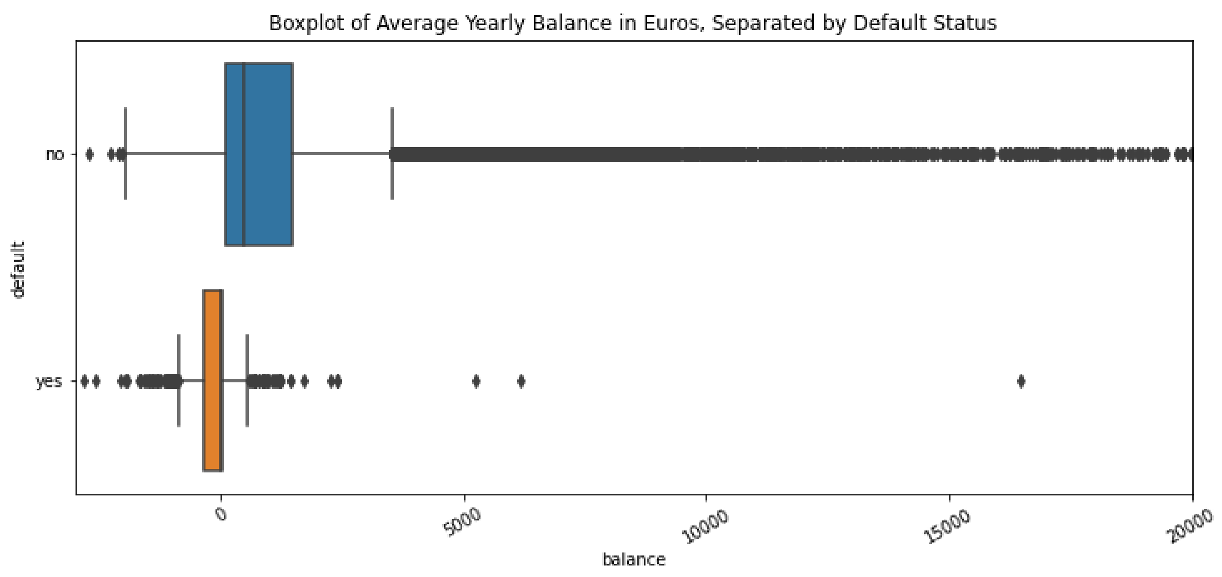
The **balance** variable is both quantitative and continuous, and it represents the potential client's average yearly balance in Euros. While the full range goes from -8019 to +102127 Euros, for visualization purposes we restrict the plots to a range of -3000 to +20000 Euros.



If we look instead at the boxplot for **balance** separated by **deposit** status, we see that the IQR is shifted to the right for those who opened a bank term deposit. In other words, people who opened a bank term deposit tend to have a higher bank balance than those who do not—and this aligns with a common sense idea that people with more money are more likely to enjoy certain financial instruments.

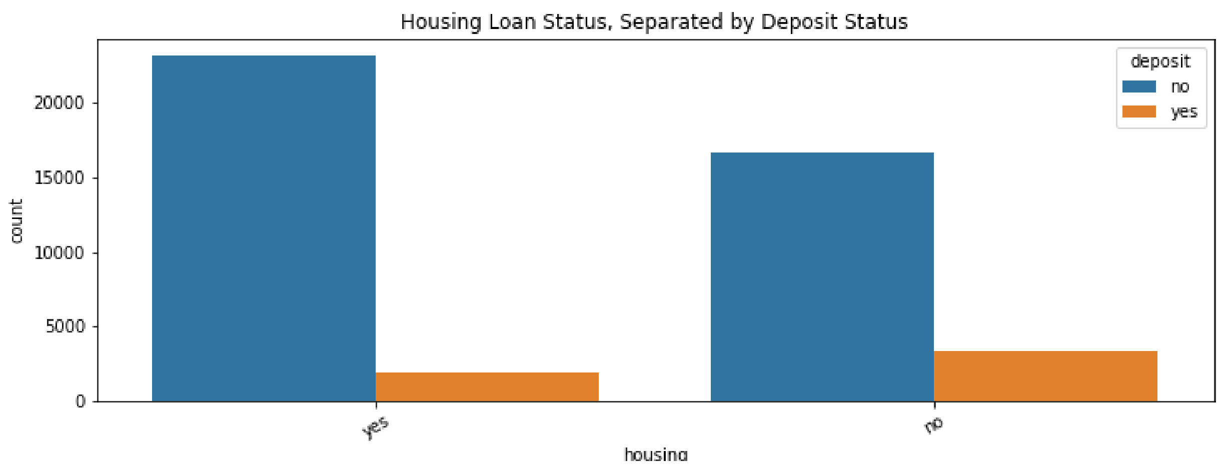


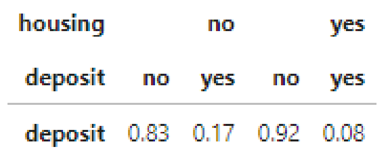
It's also worth noting that if we instead separate the boxplot by **default** status, we see that the people most likely to default tend to have a negative yearly **balance**. And this also aligns with a common sense notion, that people who owe money are more likely to default on their financial obligations than people who do not owe money.



Housing

When looking at the binary variable for **housing**, it seems like a true statement to say that people who do not have a **housing** loan are more than twice as likely (on a proportional basis) to agree to a bank term **deposit**.

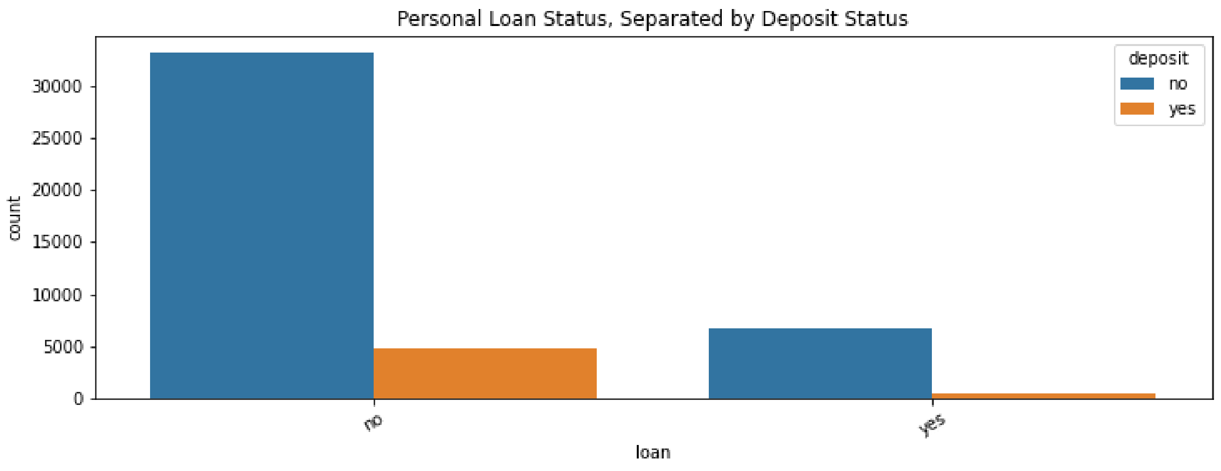




housing	no		yes	
deposit	no	yes	no	yes
deposit	0.83	0.17	0.92	0.08

Loan

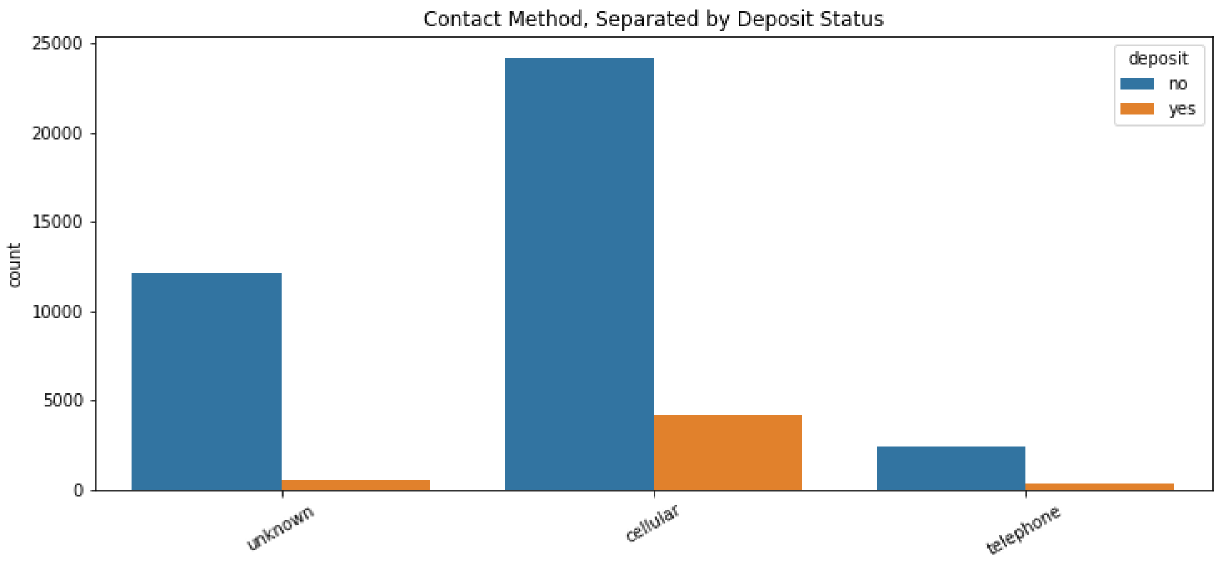
The yes/no status of a personal **loan** seems to have an impact on whether a potential client agrees to a bank term **deposit**, similar to the **housing** variable, although for personal **loan** status the binary yes/no classes are more imbalanced and thus may be more prone to small sample errors.



loan	no		yes	
deposit	no	yes	no	yes
deposit	0.87	0.13	0.93	0.07

Contact

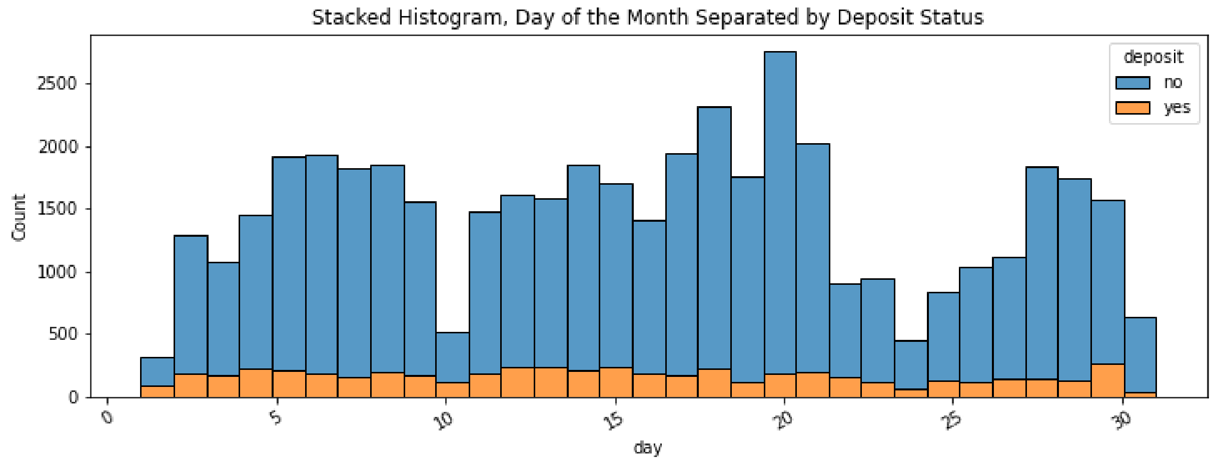
The **contact** variable consists of 3 categories, one of which is *unknown*. For those potential clients that we do know the contact method, whether *cellular* or *telephone*, there doesn't seem to be a significant difference in **deposit** status proportion rates.

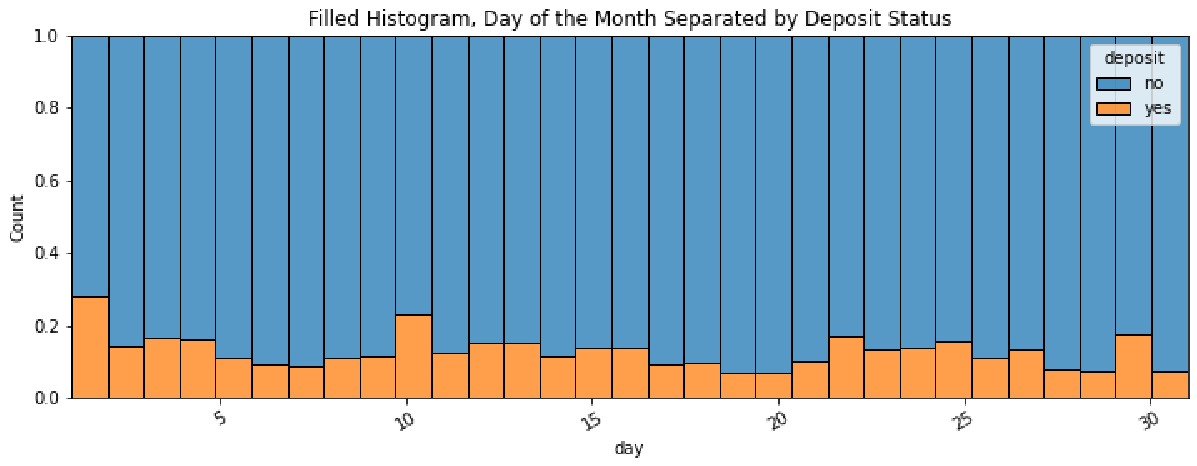


contact	cellular		telephone		unknown	
deposit	no	yes	no	yes	no	yes
deposit	0.85	0.15	0.87	0.13	0.96	0.04

<u>Day</u>

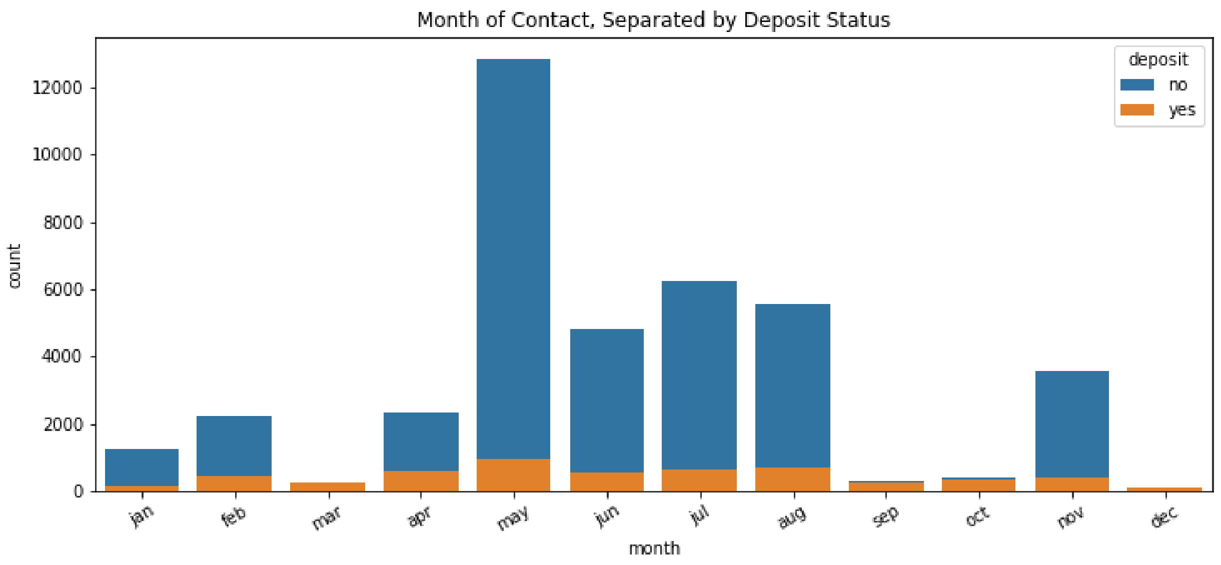
The **day** variable refers to the day of the month that a potential client was contacted, and was originally recorded as a numerical/ordinal variable. However, because of the periodic nature for days of the month, it makes sense to recode this as a categorical variable—otherwise the model may give a weight to day 20 which is 10x higher than the weight for day 2, similar to linear regression models. For additional views, we looked at stacked histograms and filled plots to get a visual sense for how **deposit** status may change with day of the month.





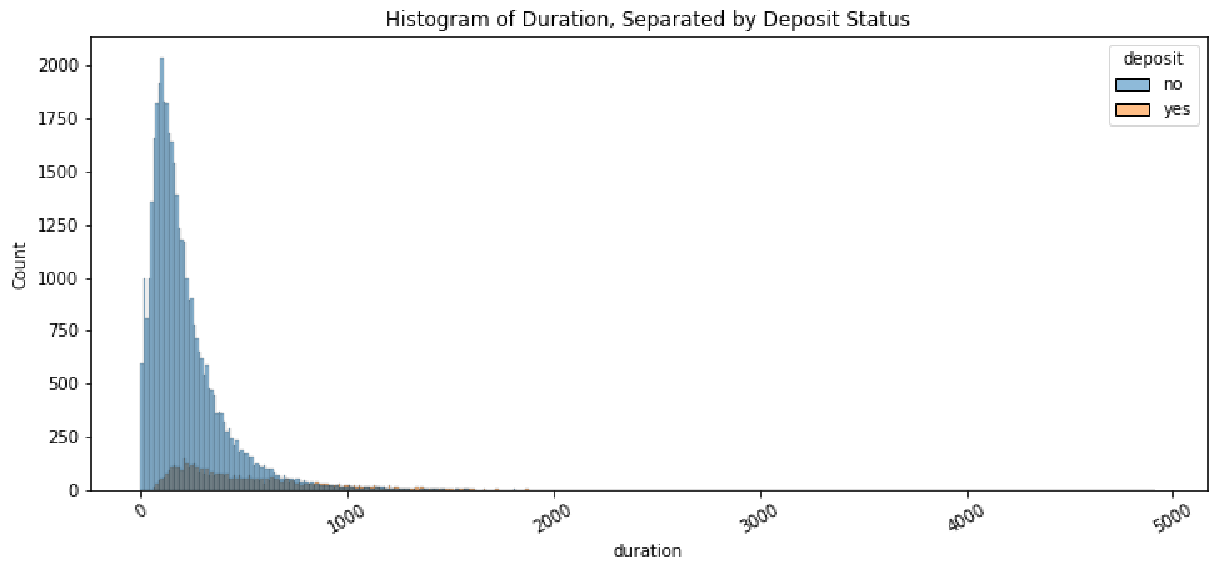
Month

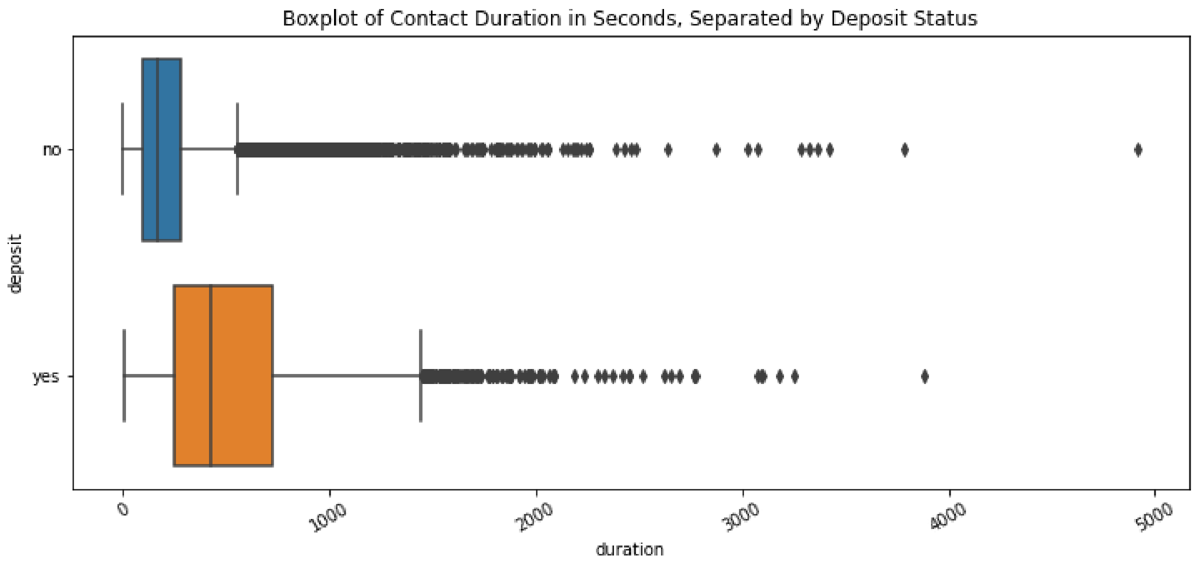
The **deposit** status varies by **month**, and indeed the **month** classes themselves are not uniformly distributed. While we do know the study took place from May 2008 to November 2010, and thus the months from May to November should be disproportionately represented, it is unknown why the actual distribution varies so widely between months and whether that affects **deposit** status.



Duration

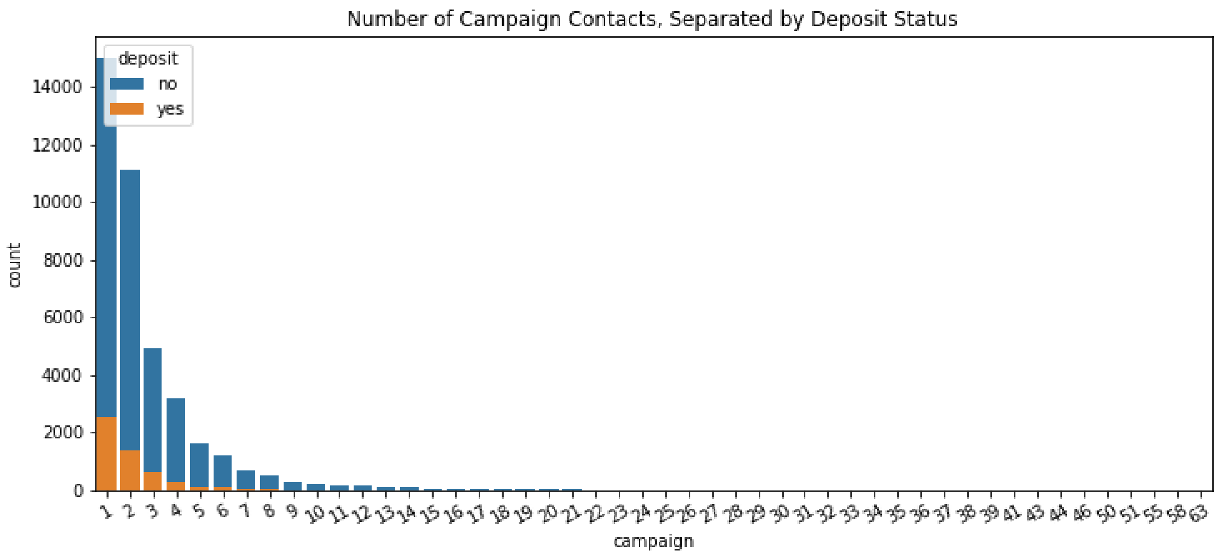
The **duration** variable is a measurement, in seconds, of how long the last contact lasted. When we look at the boxplot separated by **deposit** status, it's clear that the IQR for *yes* is shifted to longer times than the IQR for *no*, which makes sense that people who secure a bank term deposit are more likely to have a longer conversation than people who simply say no. However, for modeling purposes we'll need to be very careful with this variable—if we want to predict which customers are more likely to sign up, such that we can save resources by contacting fewer people, we may not actually know the **duration** variable before contacting potential clients.

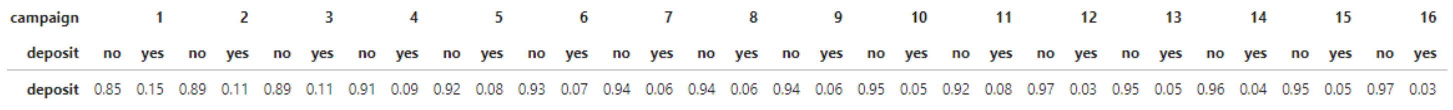




Campaign

The **contact** variable measures how many contacts were performed during a campaign for a given client. According to the data, those who agreed to a bank term **deposit** were more likely to say *yes* in the first few contacts rather than multiple contacts. Perhaps we should consider a cutoff of 15-20 contacts per person to eliminate excessive calls which drain time and resources for little return.

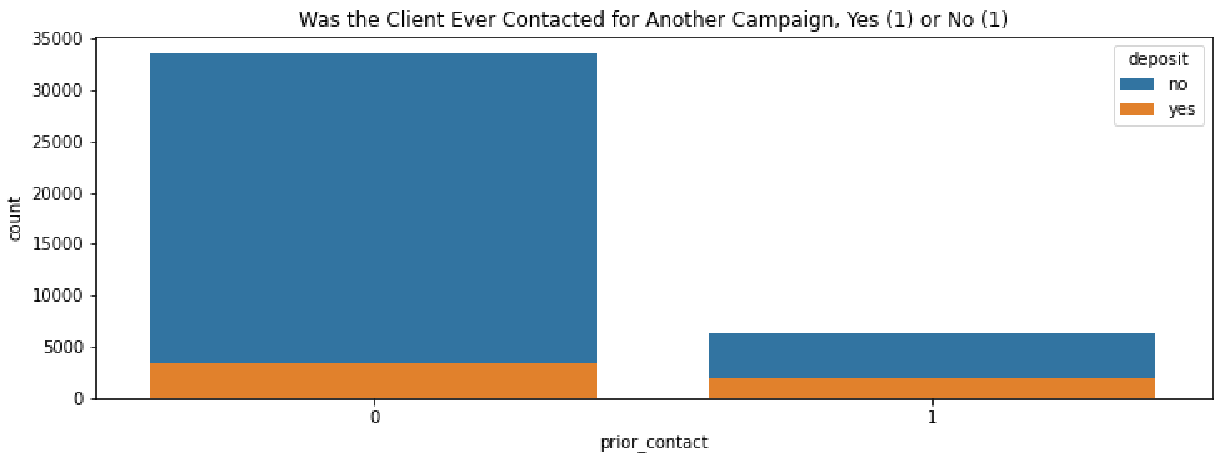




campaign	1		2		3		4		5		6		7		8		9		10		11		12		13		14		15		16	
deposit	no	yes	no	yes	no	yes	no	yes	no	yes	no	yes	no	yes	no	yes	no	yes	no	yes	no	yes	no	yes	no	yes	no	yes	no	yes	no	yes
deposit	0.85	0.15	0.89	0.11	0.89	0.11	0.91	0.09	0.92	0.08	0.93	0.07	0.94	0.06	0.94	0.06	0.94	0.06	0.95	0.05	0.92	0.08	0.97	0.03	0.95	0.05	0.96	0.04	0.95	0.05	0.97	0.03

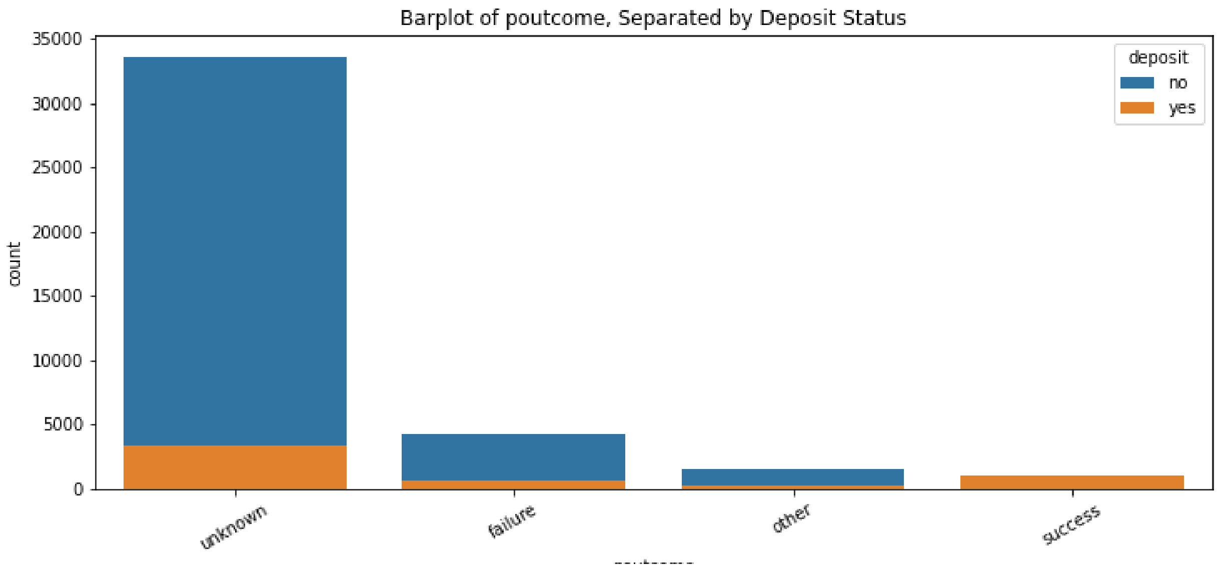
pdays and previous

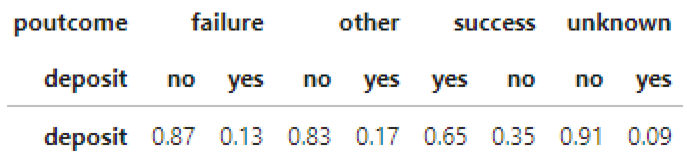
The **pdays** variable refers to the number of days that passed since the client was last contacted by a previous campaign (no contact is recorded as -1), and the **previous** variable refers to the number of contact performed before this campaign and for this client (no contact is recorded as 0). However, the count of (-1) for **pdays** is exactly the same as the count of 0 for **previous**, which is 39922 observations. Because this represents more than 75% of the recorded observations, it makes sense to combine these into a single binary variable which flags whether or not the potential client was *ever* contacted previously for a different campaign.



poutcome

The **poutcome** variable tells us the outcome of a previous marketing campaign. Because it's dominated by the *unknown* category, it's difficult to infer any actionable insights. However, there is some evidence indicating that a previous *success* has a higher proportion of landing a bank term **deposit**.





poutcome	failure		other		success		unknown	
deposit	no	yes	no	yes	yes	no	no	yes
deposit	0.87	0.13	0.83	0.17	0.65	0.35	0.91	0.09

Initial Classification Model

The primary goal of our analysis was to predict as many potential clients who would agree to a bank term deposit, and as a secondary result to minimize the total number of contacts made (because of the resource cost).

First thing we did was take the data at face value, and tried to predict **deposit** *yes* or *no* using a simple logistic regression model classifier as supplied by the sklearn package. This is a supervised learning algorithm, and the dependent (or *target*) variable is the binary yes/no for the **deposit** variable.

Data cleaning steps include binary classification on **deposit**, encoding dummy variables, and dropping null values which only came from the **age** variable.

The steps are, we define an X matrix as all of the features (other than **deposit**) and a y vector as all of the **deposit** entries. Then we split the data into training and testing sets, so we can *train* the logistic regression algorithm using the X_train and y_train data subsets. Then we make predictions on the X_test data subset, which we compare against the y_test actual results. This gives us a list of predicted values and their errors.

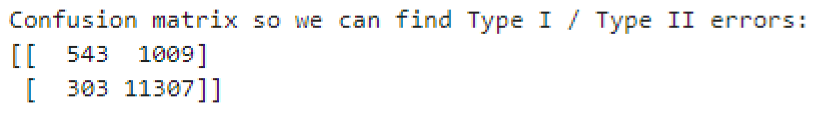
Because we're working in the space of binary classification, we can create a *confusion matrix* which gives us a total count of the following:

- True Positives
    - We guessed "yes" - 0
    - Answer is "yes" - 0
- False Positives
    - We guessed "yes" - 0
    - Answer is "no" - 1
- True Negatives
    - We guessed "no" - 1
    - Answer is "no" - 1
- False Negatives
    - We guessed "no" - 1
    - Answer is "yes" - 0

A typical confusion matrix is set up as follows:

Confusion Matrix	Predicted 0	Predicted 1
Actual 0	**True Positives**	**False Negatives**
Actual 1	**False Positives**	**True Negatives**

For example, here is a confusion matrix printout of our initial logistic regression model:



```
Confusion matrix so we can find Type I / Type II errors:
[[  543  1009]
 [  303 11307]]
```

On the basis of these four results, we can define metrics like:

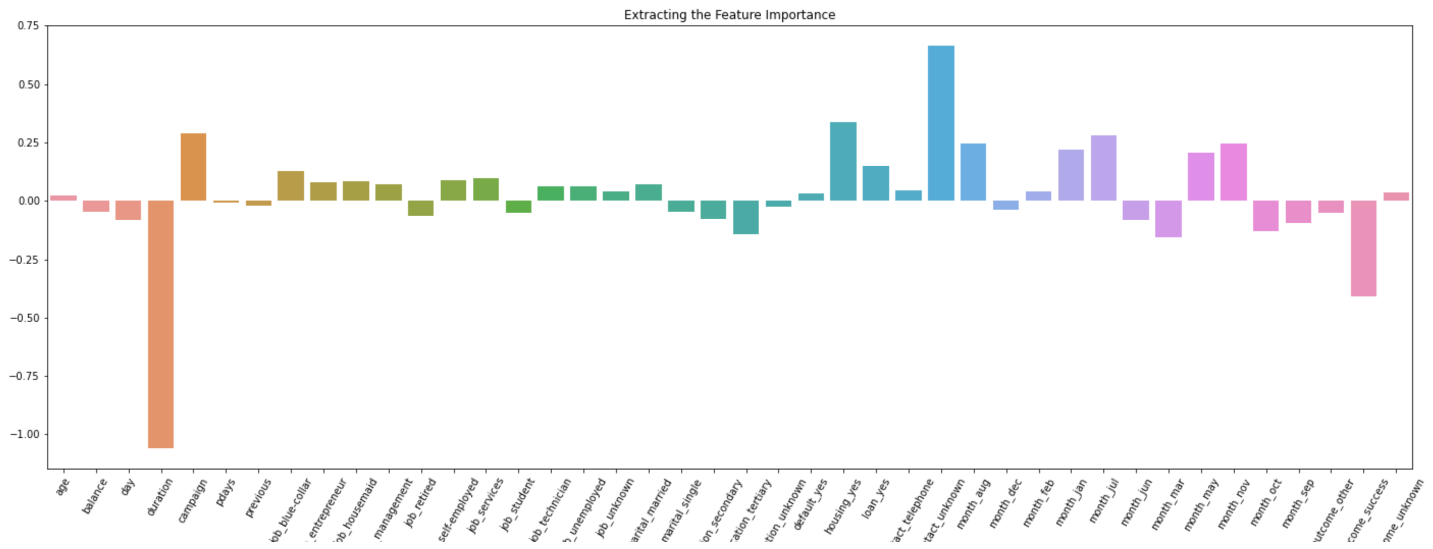
- **Sensitivity** or **Recall** or **True Positive Rate = TP / (TP + FN)**
- **Specificity** or **True Negative Rate = TN / (TN + FP)**
- **Precision = TP / (TP + FP)**

```
Here is a classification report, based on the confusion matrix
              precision    recall  f1-score   support

           0       0.64      0.35      0.45      1552
           1       0.92      0.97      0.95     11610

    accuracy                           0.90     13162
   macro avg       0.78      0.66      0.70     13162
weighted avg       0.89      0.90      0.89     13162
```

These metrics and others like ROC score will help us compare classification models against one another.

Logistic Regression Feature Importance (before data transformation)

After running our first logistic regression model, we looked at the *feature importances* to get a sense for what the model considered to be important. We then considered PCA or other dimensionality reduction techniques, but ultimately decided to engineer features directly in the ways outlined in the Preprocessing section. The following bar chart visually shows the magnitude of regression coefficients for each feature in the initial linear regression model.



Preprocessing

Imputing Missing Values

The only variable that had missing values was the age variable, which contained 1339 missing observations. We imputed the missing values with an Iterative Imputer, which is a "multivariate imputer that estimates each feature from all the others. A strategy for imputing missing values by modeling each feature with missing values as a function of other features in a round-robin fashion."
Reference: https://scikit-learn.org/stable/modules/generated/sklearn.impute.IterativeImputer.html

Recoding data

The first transformation we made was to recode our target variable **deposit** as a binary indicator, 0 for "yes" deposit and 1 for "no" deposit. In this way we can treat our output predictions as binary classifications, and use the relevant analytical tools such True/False Positives, Type I and II errors, confusion matrix, specificity/accuracy, and to compare performances between models.

As discussed in the EDA section, we chose to recode **pdays** and **previous** as a binary variable called **prior_contact**, flagging whether or not the potential client was *ever* previously contacted.

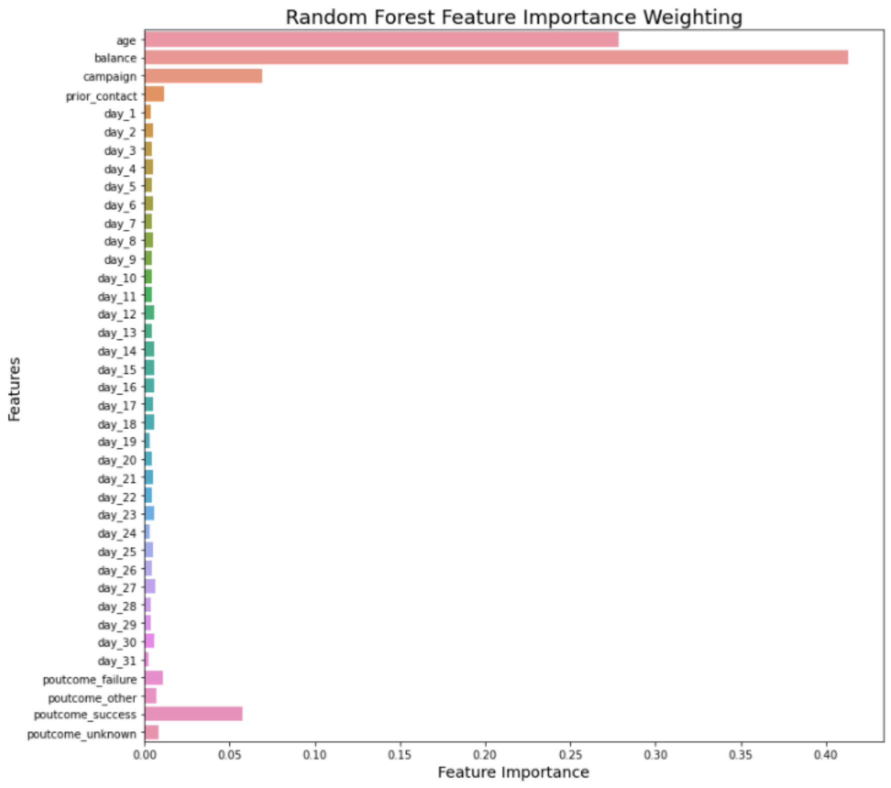
We also transformed the **days** integer variable into a categorical variable, as discussed in the EDA section.

Scaling Data - MinMax, Standard, Robust

In order to properly use machine learning models, we needed to normalize or "scale" the data. The most commonly used method is a Standard scalar, but we also decided to explore a MinMax scaler and a Robust scaler. The Standard scaler is used for normal distributions, where each feature is to have a zero-mean, unit standard-deviation. This scales the data to unit variance. The MinMax scaler scales all the features in the range of [0,1]. It scales the values to a specific value range without changing the shape of the original distribution. The robust scalar is like the MinMax scalar, but it is more robust to outliers, since it uses the interquartile range instead of the minimum and maximum. We assessed the performance of the classifiers comparatively when using each of the scalars in order to decide which was best to use for our final model.

Random Forest Feature Importance (after data transformation)

Once we re-coded and transformed the data, as discussed in the pre-processing section, we assessed feature importances with the random forest classifier. We tried dropping multiple features that had a low feature importances such as the **day** (of the month), **prior_contact**, **poutcome**, and even **campaign**, but dropping them all resulted in a worse performance of our model.



Machine Learning Models (Classifiers)

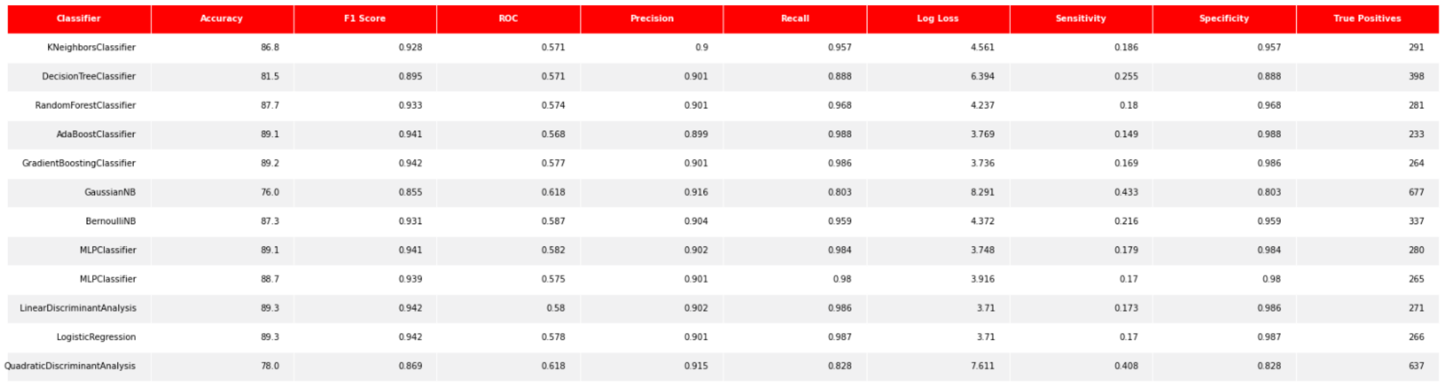
We decided to explore the output from several different classification algorithms. To do this efficiently, we looped through 12 classifiers (shown below) and assessed each one's performance using metrics such as : Accuracy, F1 score, ROC score, Precision, Recall, Log Loss, Sensitivity, Specificity and total count for true positives.

Our primary goal towards choosing a final model to use, was maximizing true positives while secondarily minimizing false negatives. Since there is an imbalance of data in our target variable (deposit), we could not only rely on accuracy as a measure of performance, since the algorithm will easily have a high accuracy since 88.31% of the data are "no deposit". Therefore, we primarily looked at metrics such as recall, ROC score, sensitivity and specificity.

Precision and recall focus on the True Positives, while Sensitivity and Specificity focus on the correct predictions. In our case, we decided to focus more on maximizing the amount of true positives, which led us to rely on sensitivity and the true positive count. Having a balance between these metrics are also very important, which is what the ROC score does for us.

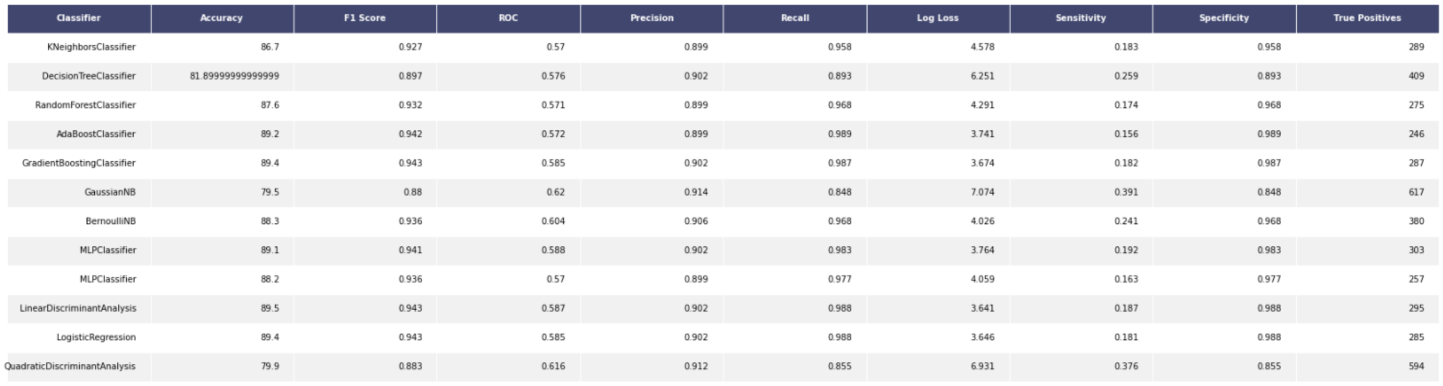
Comparison of Different Scalers and Respective Classifier Performance

MinMaxScaler



Classifier	Accuracy	F1 Score	ROC	Precision	Recall	Log Loss	Sensitivity	Specificity	True Positives
KNeighborsClassifier	86.8	0.928	0.571	0.9	0.957	4.561	0.186	0.957	291
DecisionTreeClassifier	81.5	0.895	0.571	0.901	0.888	6.394	0.255	0.888	398
RandomForestClassifier	87.7	0.933	0.574	0.901	0.968	4.237	0.18	0.968	281
AdaBoostClassifier	89.1	0.941	0.568	0.899	0.988	3.769	0.149	0.988	233
GradientBoostingClassifier	89.2	0.942	0.577	0.901	0.986	3.736	0.169	0.986	264
GaussianNB	76.0	0.855	0.618	0.916	0.803	8.291	0.433	0.803	677
BernoulliNB	87.3	0.931	0.587	0.904	0.959	4.372	0.216	0.959	337
MLPClassifier	89.1	0.941	0.582	0.902	0.984	3.748	0.179	0.984	280
MLPClassifier	88.7	0.939	0.575	0.901	0.98	3.916	0.17	0.98	265
LinearDiscriminantAnalysis	89.3	0.942	0.58	0.902	0.986	3.71	0.173	0.986	271
LogisticRegression	89.3	0.942	0.578	0.901	0.987	3.71	0.17	0.987	266
QuadraticDiscriminantAnalysis	78.0	0.869	0.618	0.915	0.828	7.611	0.408	0.828	637

StandardScaler



Classifier	Accuracy	F1 Score	ROC	Precision	Recall	Log Loss	Sensitivity	Specificity	True Positives
KNeighborsClassifier	86.7	0.927	0.57	0.899	0.958	4.578	0.183	0.958	289
DecisionTreeClassifier	81.89999999999999	0.897	0.576	0.902	0.893	6.251	0.259	0.893	409
RandomForestClassifier	87.6	0.932	0.571	0.899	0.968	4.291	0.174	0.968	275
AdaBoostClassifier	89.2	0.942	0.572	0.899	0.989	3.741	0.156	0.989	246
GradientBoostingClassifier	89.4	0.943	0.585	0.902	0.987	3.674	0.182	0.987	287
GaussianNB	79.5	0.88	0.62	0.914	0.848	7.074	0.391	0.848	617
BernoulliNB	88.3	0.936	0.604	0.906	0.968	4.026	0.241	0.968	380
MLPClassifier	89.1	0.941	0.588	0.902	0.983	3.764	0.192	0.983	303
MLPClassifier	88.2	0.936	0.57	0.899	0.977	4.059	0.163	0.977	257
LinearDiscriminantAnalysis	89.5	0.943	0.587	0.902	0.988	3.641	0.187	0.988	295
LogisticRegression	89.4	0.943	0.585	0.902	0.988	3.646	0.181	0.988	285
QuadraticDiscriminantAnalysis	79.9	0.883	0.616	0.912	0.855	6.931	0.376	0.855	594

RobustScaler

Classifier	Accuracy	F1 Score	ROC	Precision	Recall	Log Loss	Sensitivity	Specificity	True Positives
KNeighborsClassifier	87.3	0.931	0.569	0.901	0.963	4.37	0.176	0.963	271
DecisionTreeClassifier	81.89999999999999	0.897	0.563	0.901	0.894	6.267	0.232	0.894	358
RandomForestClassifier	88.1	0.935	0.576	0.902	0.971	4.115	0.18	0.971	278
AdaBoostClassifier	89.3	0.942	0.573	0.901	0.987	3.713	0.159	0.987	245
GradientBoostingClassifier	89.3	0.942	0.58	0.903	0.985	3.708	0.175	0.985	270
GaussianNB	77.3	0.865	0.615	0.916	0.819	7.848	0.41	0.819	632
BernoulliNB	89.3	0.942	0.581	0.903	0.985	3.685	0.178	0.985	274
MLPClassifier	88.9	0.94	0.576	0.902	0.981	3.825	0.171	0.981	264
MLPClassifier	87.3	0.931	0.582	0.904	0.959	4.382	0.205	0.959	316
LinearDiscriminantAnalysis	89.3	0.942	0.58	0.903	0.985	3.687	0.175	0.985	270
LogisticRegression	89.3	0.943	0.578	0.903	0.986	3.68	0.171	0.986	263
QuadraticDiscriminantAnalysis	79.4	0.879	0.603	0.912	0.85	7.13	0.357	0.85	550

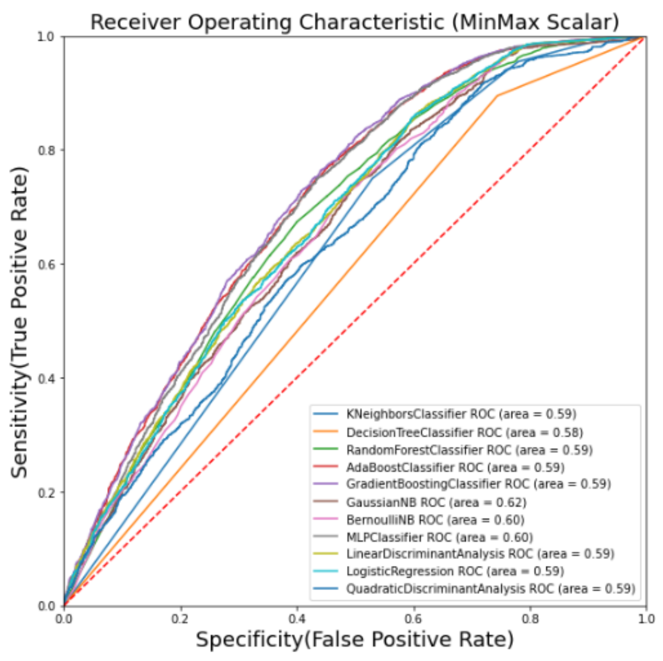
Final model selection

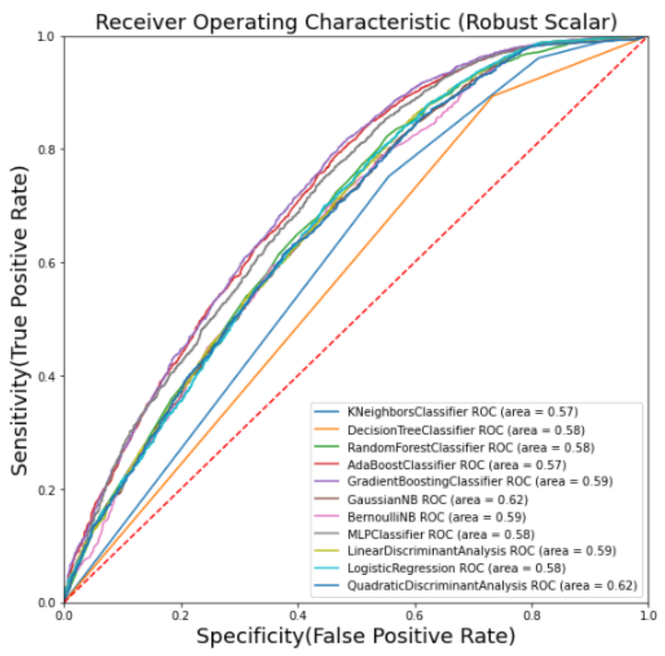
For our final model, after recoding the data as described above, we then excluded the following variables: ['pdays','previous','job','marital','education','default','housing','loan','contact','month','duration'] and we were left with

- age
- balance
- day
- campaign
- poutcome

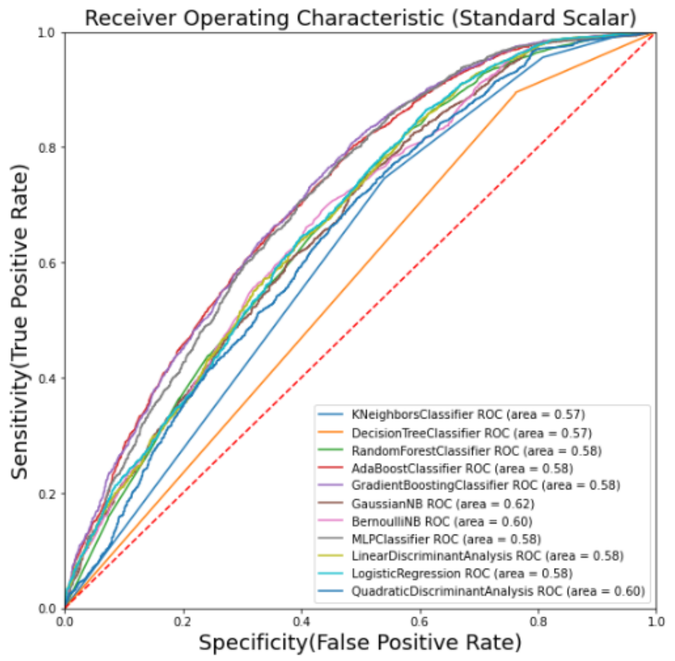
For all models, including our final model, Gaussian Naive Bayes and Quadratic Discriminant Analysis gave us the best consistent responses across different scalers and variables. They also consistently yielded the highest True Positives and Sensitivity performance, as well as best ROC score.

ROC overlay of all classifiers





Receiver Operating Characteristic (Robust Scalar)
Sensitivity(True Positive Rate)
Specificity(False Positive Rate)
0.0
0.2
0.4
0.6
0.8
1.0
KNeighborsClassifier ROC (area = 0.57)
DecisionTreeClassifier ROC (area = 0.58)
RandomForestClassifier ROC (area = 0.58)
AdaBoostClassifier ROC (area = 0.57)
GradientBoostingClassifier ROC (area = 0.59)
GaussianNB ROC (area = 0.62)
BernoulliNB ROC (area = 0.59)
MLPClassifier ROC (area = 0.58)
LinearDiscriminantAnalysis ROC (area = 0.59)
LogisticRegression ROC (area = 0.58)
QuadraticDiscriminantAnalysis ROC (area = 0.62)



Receiver Operating Characteristic (Standard Scalar)
Sensitivity(True Positive Rate)
Specificity(False Positive Rate)
0.0
0.2
0.4
0.6
0.8
1.0
KNeighborsClassifier ROC (area = 0.57)
DecisionTreeClassifier ROC (area = 0.57)
RandomForestClassifier ROC (area = 0.58)
AdaBoostClassifier ROC (area = 0.58)
GradientBoostingClassifier ROC (area = 0.58)
GaussianNB ROC (area = 0.62)
BernoulliNB ROC (area = 0.60)
MLPClassifier ROC (area = 0.58)
LinearDiscriminantAnalysis ROC (area = 0.58)
LogisticRegression ROC (area = 0.58)
QuadraticDiscriminantAnalysis ROC (area = 0.60)

Conclusions, Recommendations, Next Steps

We explored the data structure and characteristic statistics for the Portuguese Banking dataset, with an emphasis to understand how the **deposit** variable may be influenced by or predicted from the other variables. We imputed missing values, recoded information, and dropped some variables to better reflect the real-world use case: for example, dropping the **duration** variable because if we want to predict which customers are more likely to sign up, thus saving resources by contacting fewer people, we would not actually know the **duration** variable before contact.

By reviewing different classification models with different scalers, we determined that GaussianNB and Quadratic Discriminant Analysis consistently outperformed the other classification models using default parameters. Thus, our recommendation for the banking institution as of December 2010 moving forward is to screen each potential client on the basis of known information prior to contact. The algorithm will tend to over-predict the "Yes" deposits with many False Positives (and thus a lower *Precision* score), but it will also filter out a significant count of True Negatives. On this basis our call center can be more efficient and focus on those customers who are more likely to agree to a bank term deposit.

Our best algorithms, Gaussian Naive Bayes and Quadratic Discriminant Analysis increased the expectation of "Yes" from a baseline expectation of 12% to roughly 30%. For example, one set of outcomes produced 792 True Positives and 1765 False Positives, while correctly eliminating 10243 True negatives. Therefore, by focusing only on the predicted positives (True and False), we have raised the baseline expectation to roughly 30% (792 / [792 + 1765]).

Some next steps to further optimize our algorithms, fine-tuning the hyperparameters, running ensembles with multiple algorithms, and applying cross validation.

Additionally, aside from AI/ML techniques and feature engineering, we also recommend no more than 15-20 phone contacts for a given customer (see **campaign** variable), because the diminishing returns are far too low for the effort involved after the first dozen or so contacts. We can also research more demographics and increase our banking domain knowledge to better find opportunities for continued improvement.