Module 3 – Logistic Regression

Objectives

At the end of this module, you should be able to:

- Build Logistic Regression
- Understand the concept of Maximum Likelihood Estimation
- Decide Model Evaluation Parameters
- Regularization

What is Logistic Regression?

Logistic Regression is a technique used for classification algorithm that models the probability of the output class.

- It estimates relationship between a dependent variable (target/label) and one or more independent variable (predictors) where dependent variable is categorical.
- Logistic regression in a summary is nothing but applying a sigmoidal function on the output of Linear Regression which will convert the predicted value into a value between 0 to 1. Then taking 0.5 a threshold and if the output is less than 0.5, the sample is in class 0 otherwise in class 1.

Features –

- Expects a "smooth" linear relationship with predictors.
- Logistic Regression is concerned with probability of a discrete outcome.
- Slightly less prone to over-fitting
- Because fits a shape, might work better when less data available.

Assumptions

- We apply logistic Regression only when the dependent variable is a categorical attribute.
- Generally binary logistic regression requires the dependent variable to be binary.
- For a binary regression, the factor level 1 of the dependent variable should represent the desired outcome.
- Only the meaningful variables should be included which has highest correlation with the dependent variable.
- The independent variables are linearly related to the log odds.
- Logistic regression requires quite large sample sizes.
- The independent variables should be independent of each other. That is, the model should have little or no multicollinearity.

Objective of Logistic Regression

Binary Classification:

- Given the subject and the email text predicting, Email Spam or not.
- Sunny or rainy day prediction, using the weather information.
- Based on the bank customer history, Predicting whether to give the loan or not.

Multi-Classification:

- Given the dimensional information of the object, Identifying the shape of the object.
- Identifying the different kinds of vehicles.
- Based on the color intensities, Predicting the color type

Logistic Regression – Diabetes Example

Consider

Y axis

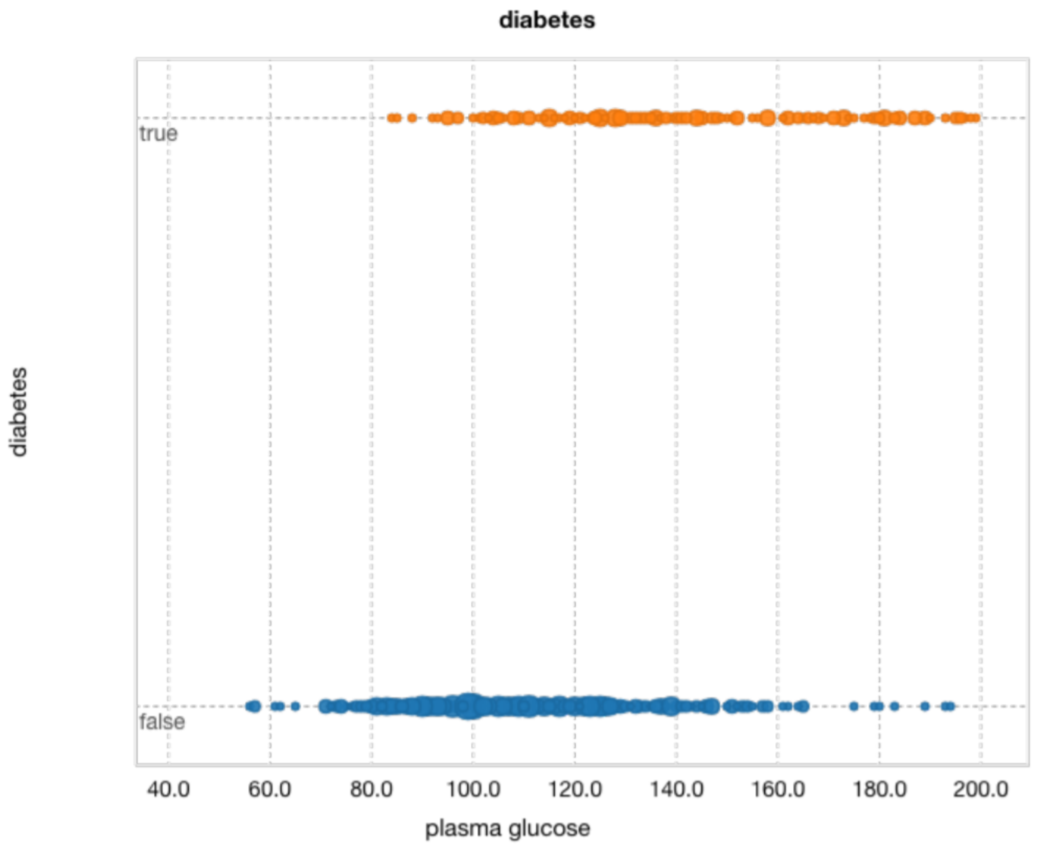
True: Perosn has diabetes

Flase: No diabetes

X Axis

Feature – plasma glucose

Plotting the data in following way, Now in order to build a classification model we will make some modification to the Linear Regression model we have seen prior.



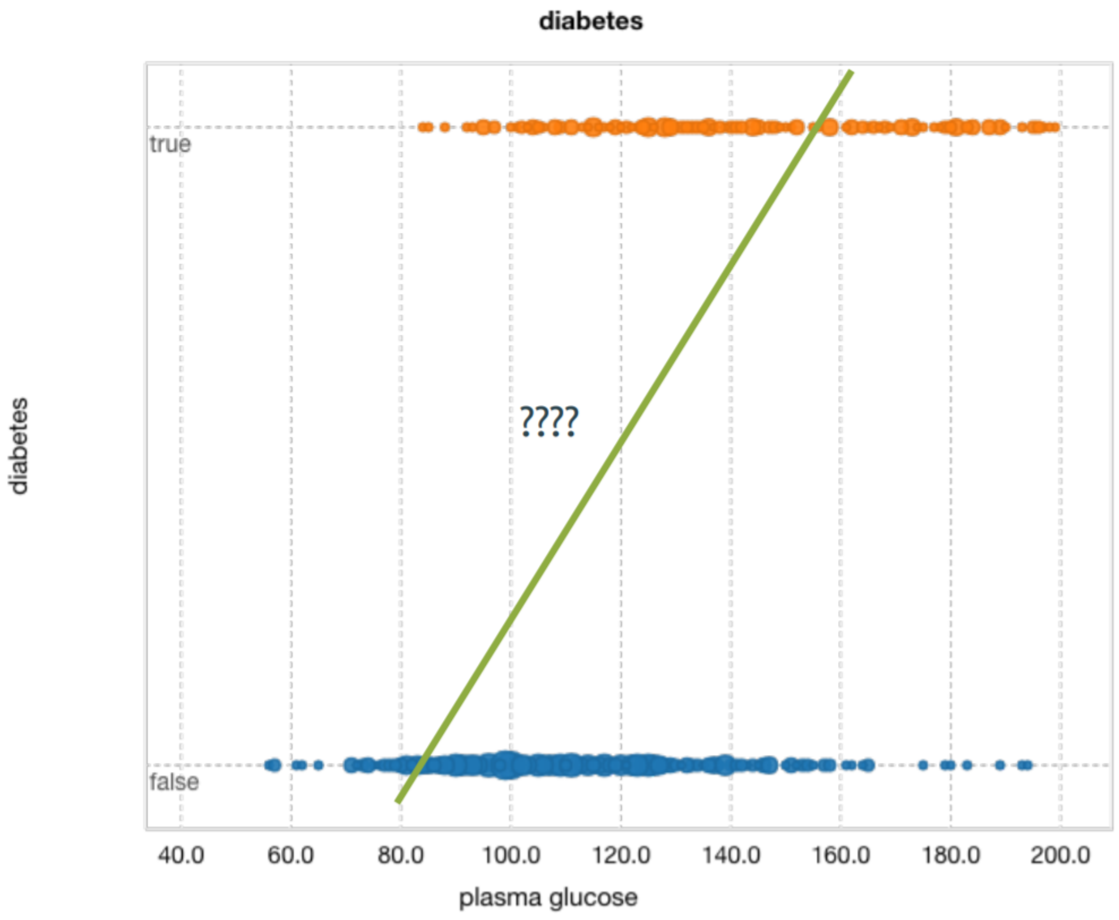
On the given data, drawing a best fit trained regression line with equation

$$\hat{y} = mx + c$$

where

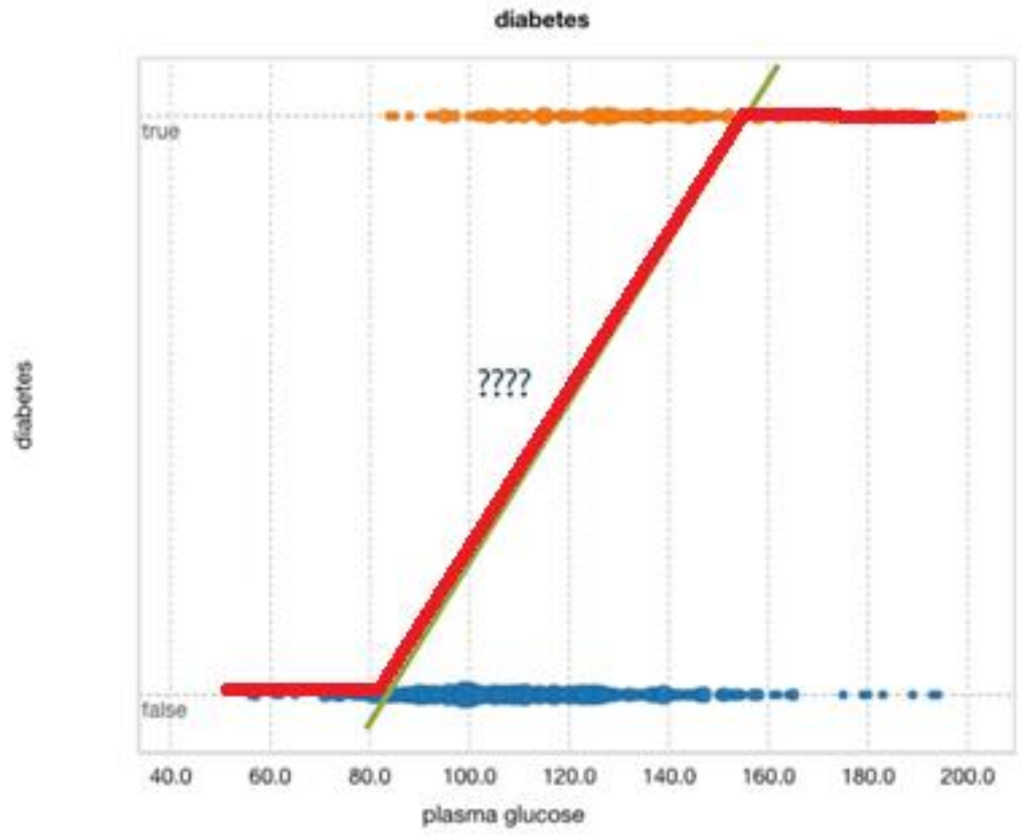
$\hat{y} = Value\ predicted\ by\ current\ Algorithm$

This will make the data look like following -



Now in this scenario if we can convert this this line into a curve between 0 to 1 we can consider the output of the line as the probability of sample belonging to class 0 or class 1.

Then taking 0.5 as a threshold it would be very easy to make classification based on the best fit line given by the Linear Regression model.



The above shape best matches with the sigmoidal function.

$$\boldsymbol{p = \frac{1}{1+e^{-\hat{y}}}}$$

Applying the above function on the predicted output from Linear Regression.

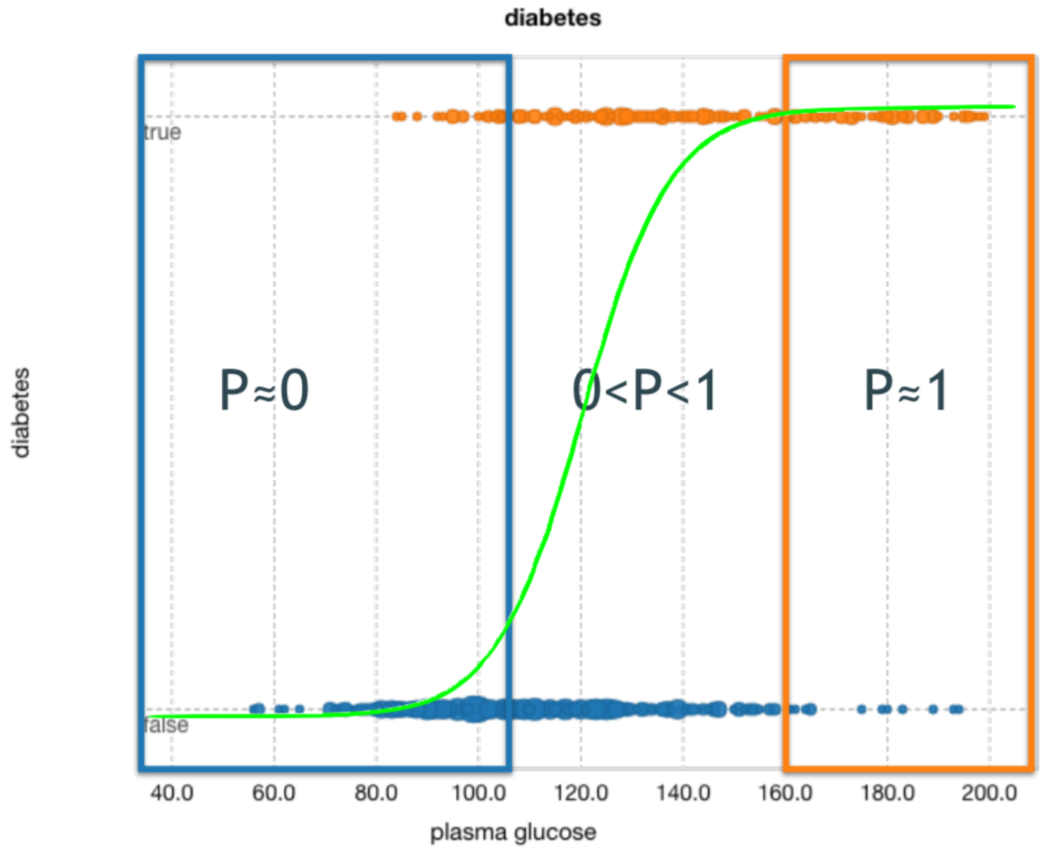
$$\Rightarrow \boldsymbol{\hat{y} = mx + c}$$

$$\Rightarrow \boldsymbol{p = \frac{1}{1+e^{-\hat{y}}}}$$

$$\Rightarrow \boldsymbol{p = \frac{1}{1+e^{-(mx+c)}}}$$

$$\Rightarrow \boldsymbol{ln\left(\frac{p}{1-p}\right) = mx + c}$$

So the final curve will look like following one –



Thus by applying some function on the top of Linear Regression model makes it a classification model which is known as Logistic Regression.

We generally address 2 class classification problem using logistic Regression but many researchers also use Softmax function as well for multiclass classification and call it Logistic Regression.

For Binary Classification using Logistic Regression

$Sigmoidal\ Function$

$$\boldsymbol{\sigma(z) = \frac{1}{1 + e^{-z}}}$$

For Multinomial Classification using Logistic Regression

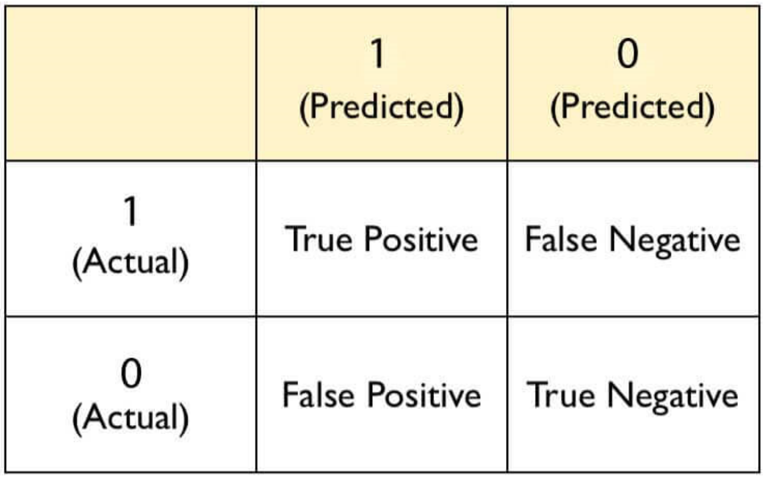
$Softmax\ function$

$$\boldsymbol{\sigma(z)_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}}\ for\ j = 1, 2, \ldots\ldots K}$$

Logistic Regression Performance Analysis

Confusion Matrix

Confusion matrix is the most crucial metric commonly used to evaluate classification models. It's quite confusing but make sure you understand it by heart. If you still don't understand anything, ask me in comments.



	1 (Predicted)	0 (Predicted)
1 (Actual)	True Positive	False Negative
0 (Actual)	False Positive	True Negative

- True positive = correctly identified
- False positive = incorrectly identified
- True negative = correctly rejected
- False negative = incorrectly rejected

Example

- True positive: Sick people correctly identified as sick
- False positive: Healthy people incorrectly identified as sick
- True negative: Healthy people correctly identified as healthy
- False negative: Sick people incorrectly identified as healthy

As you can see, the confusion matrix avoids "confusion" by measuring the actual and predicted values in a tabular format. In table above, Positive class = 1 and Negative class = 0. Following are the metrics we can derive from a confusion matrix:

Accuracy - It determines the overall predicted accuracy of the model. It is calculated as Accuracy = (True Positives + True Negatives)/(True Positives + True Negatives + False Positives + False Negatives)

$$\boldsymbol{Accuracy = \frac{TP + TN}{TP + TN + FP + FN}}$$

Recall or True Positive Rate (TPR) –

The fraction of relevant instances that have been retrieved over the total amount of relevant instances.

The recall is intuitively the ability of the classifier to find all the positive samples.

Recall indicates how many positive values, out of all(total actual) the positive values, have been correctly predicted. The formula to calculate the true positive rate is (TP/TP + FN). Also, TPR = 1 - False Negative Rate. It is also known as Sensitivity or Recall.

$$\boldsymbol{Recall\ (TPR) = \frac{TP}{TP + FN}}$$

False Positive Rate (FPR) - It indicates how many negative values, out of all the negative values, have been incorrectly predicted. The formula to calculate the false positive rate is (FP/FP + TN). Also, FPR = 1 - True Negative Rate.

$$\boldsymbol{FPR = \frac{FP}{FP + TN}}$$

True Negative Rate (TNR) - It indicates how many negative values, out of all the negative values, have been correctly predicted. The formula to calculate the true negative rate is (TN/TN + FP). It is also known as Specificity.

$$\boldsymbol{TNR = \frac{TN}{TN + FP}}$$

False Negative Rate (FNR) - It indicates how many positive values, out of all the positive values, have been incorrectly predicted. The formula to calculate false negative rate is (FN/FN + TP).

$$\boldsymbol{FNR = \frac{FN}{FN + TP}}$$

Precision: It indicates how many values, out of all the predicted positive values, are actually positive. It is formulated as:(TP / TP + FP).

$$\boldsymbol{Precision = \frac{TP}{TP + FP}}$$

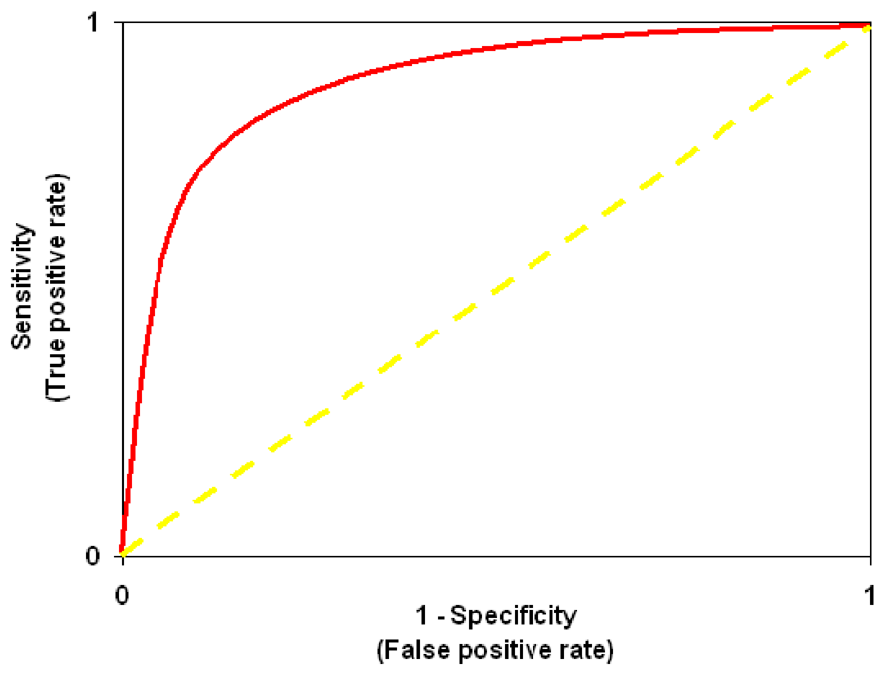
F1 Score: F1 score is the harmonic mean of precision and recall. It lies between 0 and 1. Higher the value, better the model. It is formulated as 2((precision*recall) / (precision+recall)).

$$\boldsymbol{F1\ Score = 2 * \frac{(precision * recall)}{(precision + recall)}}$$

Receiver Operator Characteristic (ROC)

ROC is used to determine the accuracy of a classification model at a user defined threshold value. It determines the model's accuracy using Area Under Curve (AUC). The area under the curve (AUC), also referred to as index of accuracy (A) or concordant index, represents the performance of the ROC curve. Higher the area, better the model. ROC is plotted between True Positive Rate (Y axis) and False Positive Rate (X Axis).

In this plot, our aim is to push the red curve (shown below) toward 1 (left corner) and maximize the area under curve. Higher the curve, better the model. The yellow line represents the ROC curve at 0.5 threshold. At this point, sensitivity = specificity.

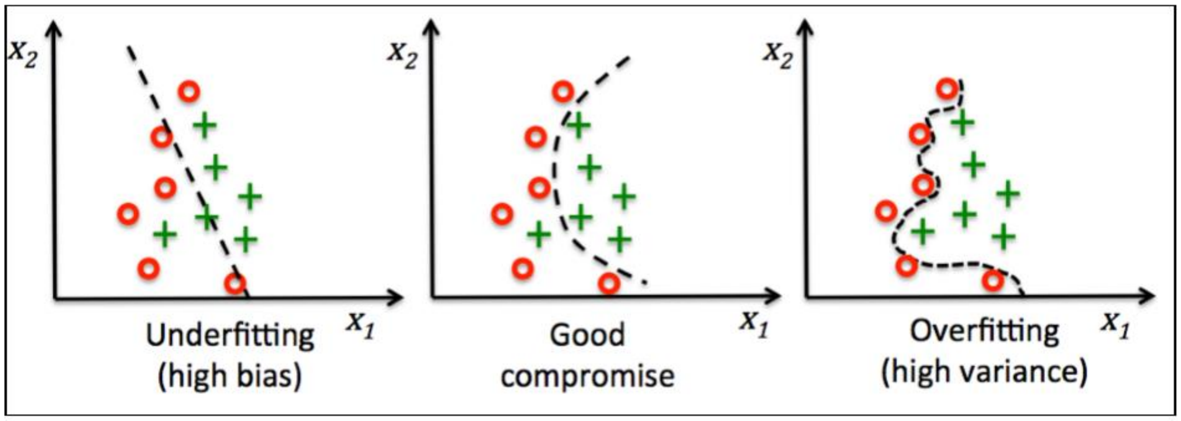


Overfitting & Generalization

Overfitting is a common problem in machine learning, where a model performs well on training data but does not generalize well to unseen data (test data).

Overfitting occurs when a model is excessively complex, such as having too many parameters relative to the number of observations.

A model that has been overfit has poor predictive performance, as it overreacts to minor fluctuations in the training data.



How to avoid overfitting?

- One of the ways to combat over-fitting is to increase the training data size.
- Another way to combat over-fitting is to perform early stopping.

  Early stopping rules provide guidance as to how many iterations can be run before the learner begins to over-fit. Early stopping rules have been employed in many different machine learning methods, with varying amounts of theoretical foundation

Early stopping based on cross-validation combats over-fitting by monitoring the model's performance on a validation set. The error on the validation set is used as a proxy for the generalization error in determining when over-fitting has begun.

- Bias represents how the model represent the training data and variance is how the model responds to variation in data.

  Variance is large is different training sets give rise to very different classifier. Variance is small if training set has minor effect on the classification decisions. Variance measures how inconsistent the decisions and essentially variation of prediction of learned classifier.

  Models with high variances are sensitive to noise, small changes in training data sets can lead to significantly different model parameters.

  Models with high bias produce simple models that do not tend to over fit but may under-fit the data failing to capture the variations in the data.Models with low bias are complex enabling them to represents the training data accurately .

- A way to combat over-fitting is through regularization. Regularization techniques can be viewed as imposing certain prior distribution on the model parameters.Mathematically the regularization process implies performing constrained optimization.

  L2 regularization implies imposing Gaussian prior on weights while L1 prior implies imposing Laplacian prior on weights.

Exercise 1 – Bank Marketing Dataset

The dataset is taken from

http://archive.ics.uci.edu/ml/datasets/Bank+Marketing

it is related to direct marketing campaigns (phone calls) of a Portuguese banking institution. The classification goal is to predict whether the client will subscribe (1/0) to a term deposit (variable y).

The dataset provides the bank customers' information. It includes 41,188 records and 21 fields.

Input variables: # bank client data:

1 - age (numeric)

2 - job : type of job (categorical: 'admin.','blue-collar', 'entrepreneur', 'housemaid', 'management', 'retired', 'self-employed', 'services', 'student', 'technician', 'unemployed', 'unknown')

3 - marital : marital status (categorical: 'divorced','married','single','unknown'; note: 'divorced' means divorced or widowed)

4 - education (categorical: 'basic.4y','basic.6y','basic.9y','high.school','illiterate','professional.course','university.degree','unknown')

5 - default: has credit in default? (categorical: 'no','yes','unknown')

6 - housing: has housing loan? (categorical: 'no','yes','unknown')

7 - loan: has personal loan? (categorical: 'no','yes','unknown')

# related with the last contact of the current campaign:

8 - contact: contact communication type (categorical: 'cellular','telephone')

9 - month: last contact month of year (categorical: 'jan', 'feb', 'mar', ..., 'nov', 'dec')

10 - day_of_week: last contact day of the week (categorical: 'mon','tue','wed','thu','fri')

11 - duration: last contact duration, in seconds (numeric).

Important note: this attribute highly affects the output target (e.g., if duration=0 then y='no'). Yet, the duration is not known before a call is performed. Also, after the end of the call y is obviously known. Thus, this input should only be included for benchmark purposes and should be discarded if the intention is to have a realistic predictive model.

12 - campaign: number of contacts performed during this campaign and for this client (numeric, includes last contact)

13 - pdays: number of days that passed by after the client was last contacted from a previous campaign (numeric; 999 means client was not previously contacted)

14 - previous: number of contacts performed before this campaign and for this client (numeric)

15 - poutcome: outcome of the previous marketing campaign (categorical: 'failure','nonexistent','success')

# social and economic context attributes
16 - emp.var.rate: employment variation rate - quarterly indicator (numeric)

17 - cons.price.idx: consumer price index - monthly indicator (numeric)

18 - cons.conf.idx: consumer confidence index - monthly indicator (numeric)

19 - euribor3m: euribor 3 month rate - daily indicator (numeric)

20 - nr.employed: number of employees - quarterly indicator (numeric)

Output variable (desired target):
21 - y - has the client subscribed a term deposit? (binary: 'yes','no')

Import the libraries

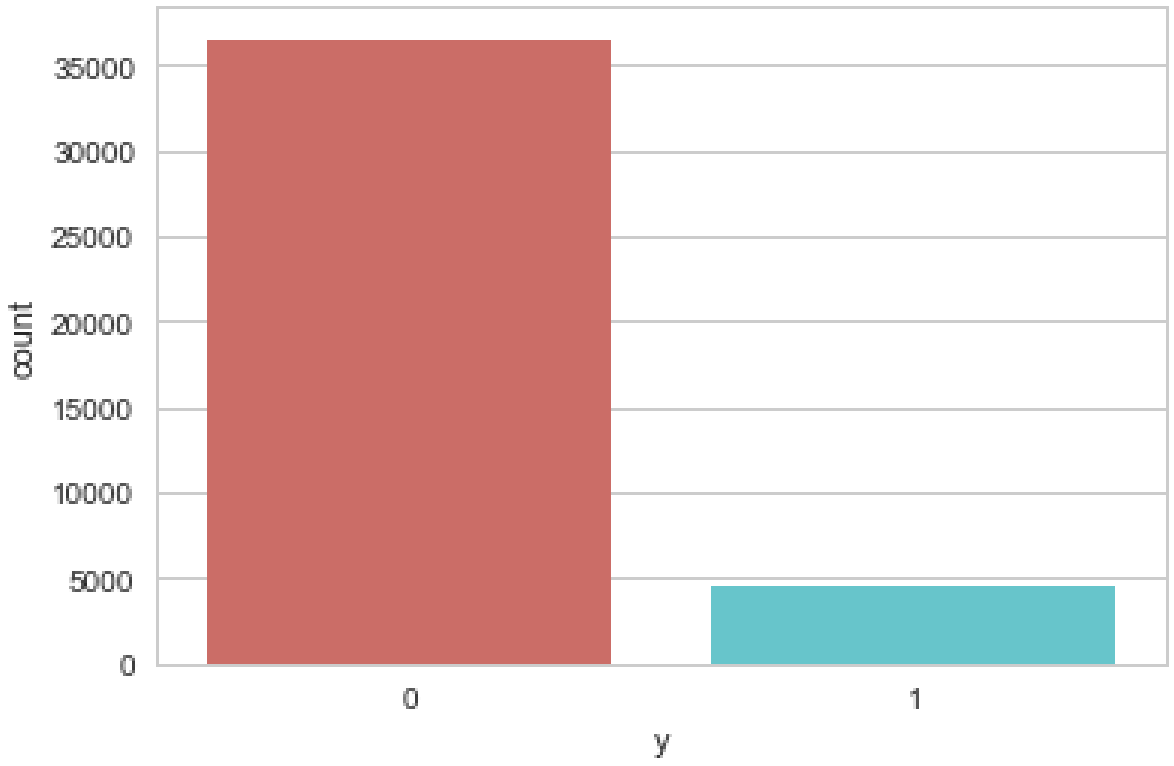
```
import pandas as pd
import numpy as np
from sklearn import preprocessing
import matplotlib.pyplot as plt
plt.rc("font", size=14)
from sklearn.linear_model import LogisticRegression
from sklearn.cross_validation import train_test_split
import seaborn as sns
sns.set(style="white")
sns.set(style="whitegrid", color_codes=True)
```

Import dataset

```
data = pd.read_csv('bank.csv', header=0)

data = data.dropna()

print(data.shape)

print(list(data.columns))
```

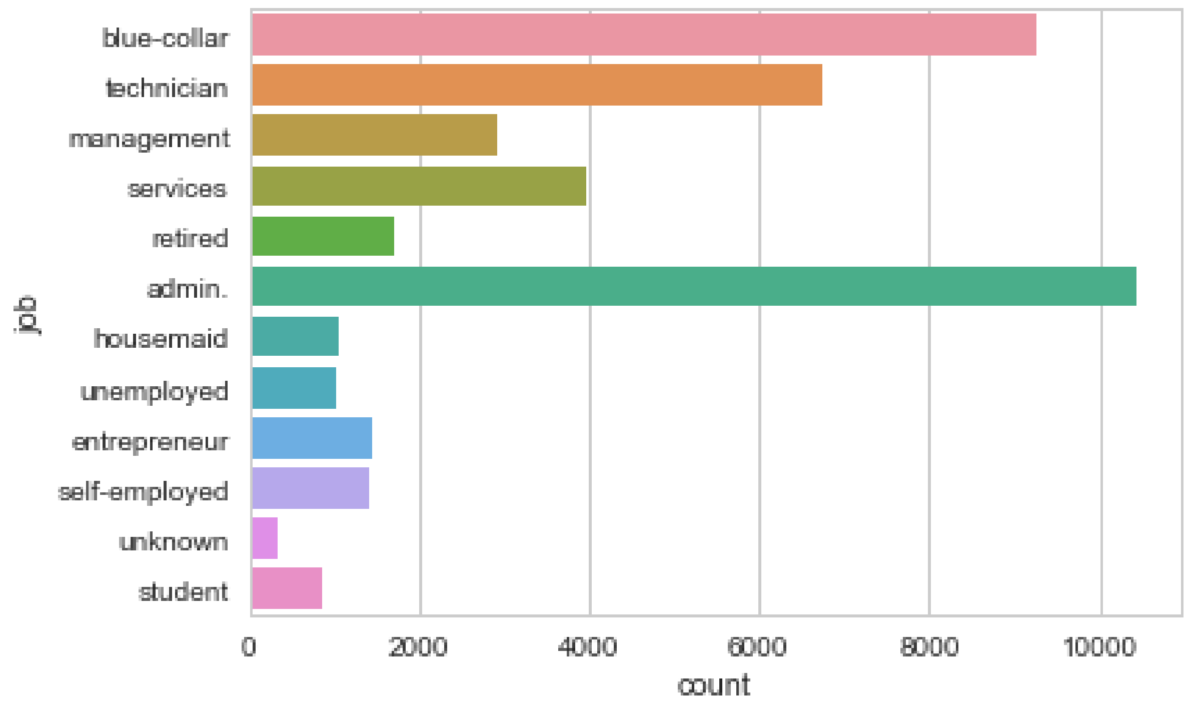
Barplot for dependent variable

```
sns.countplot(x='y',data=data, palette='hls')

plt.show()
```



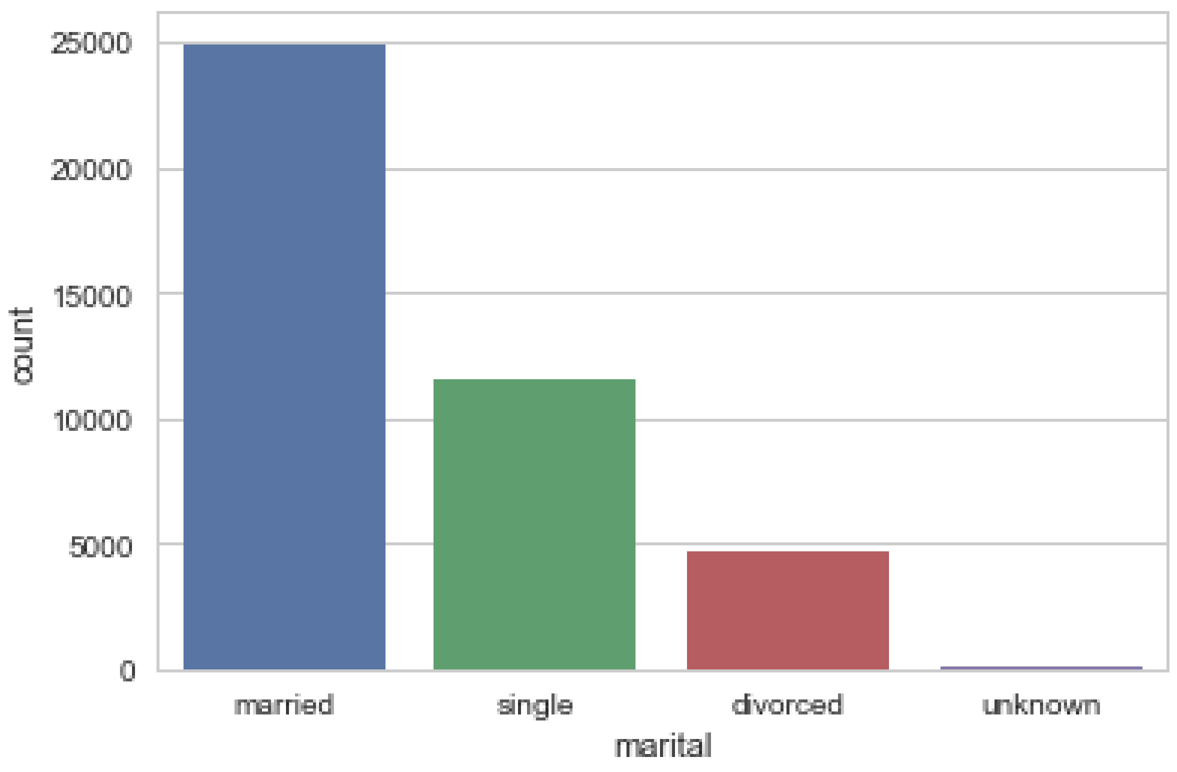
Customer job distribution

```
sns.countplot(y="job", data=data)

plt.show()
```



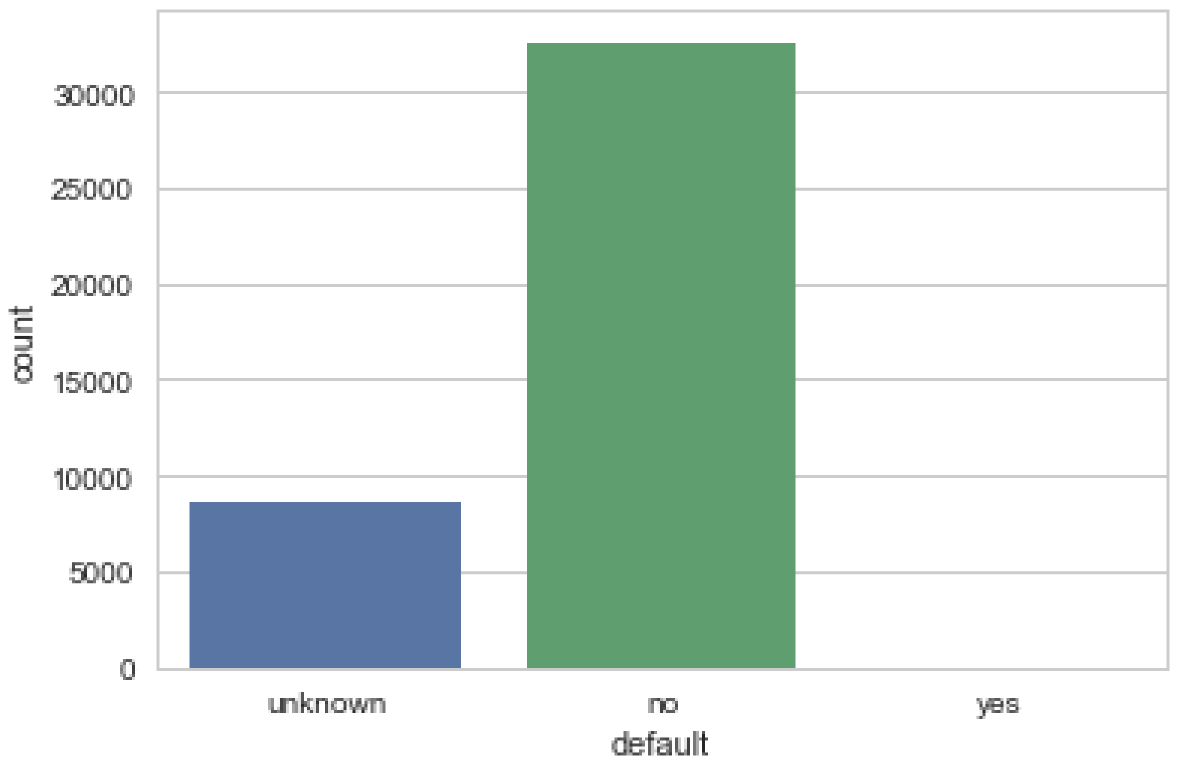
Customer Marital Status Distribution

```
sns.countplot(x="marital", data=data)

plt.show()
```



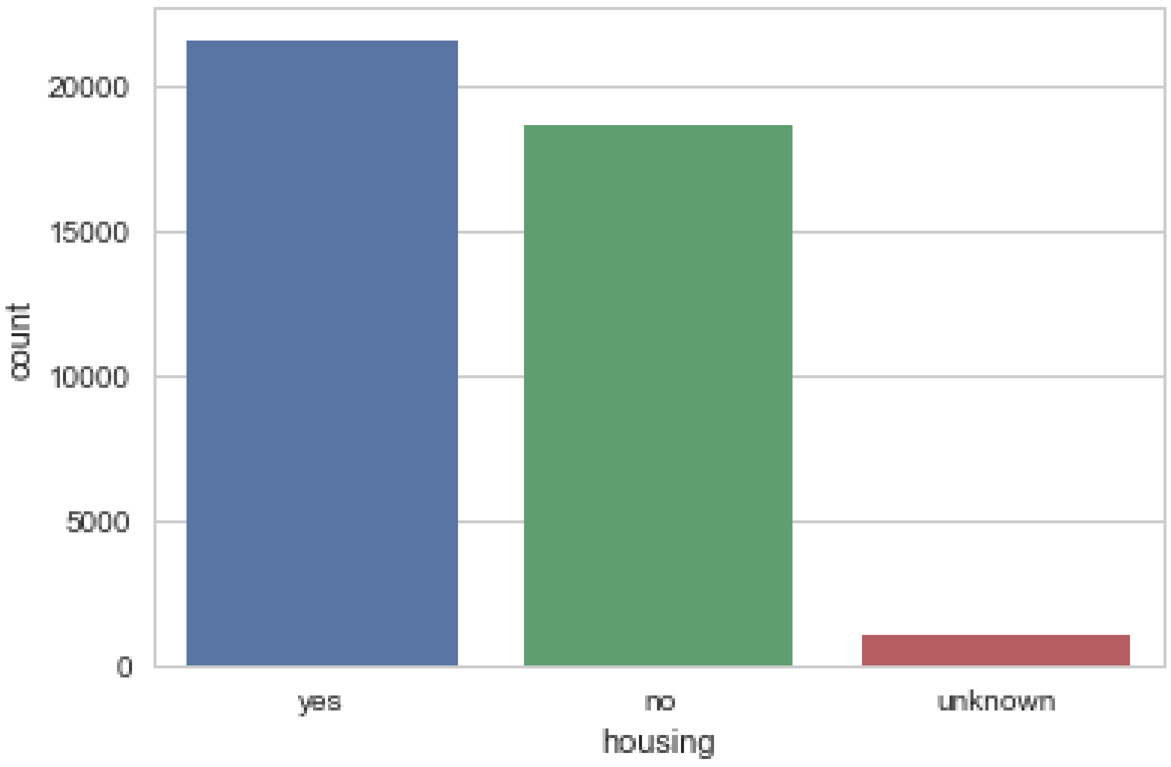
Barplot for Credit in default

```
sns.countplot(x="default", data=data)

plt.show()
```



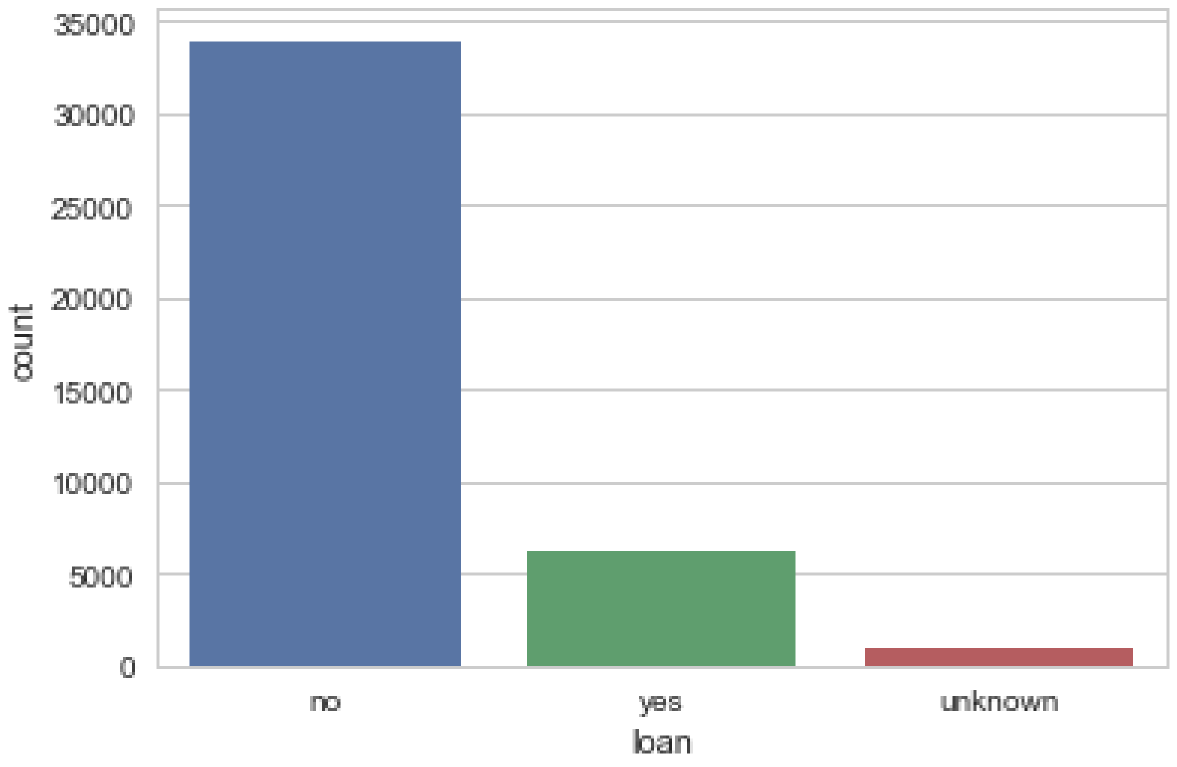
Barplot for housing loan

```
sns.countplot(x="housing", data=data)

plt.show()
```



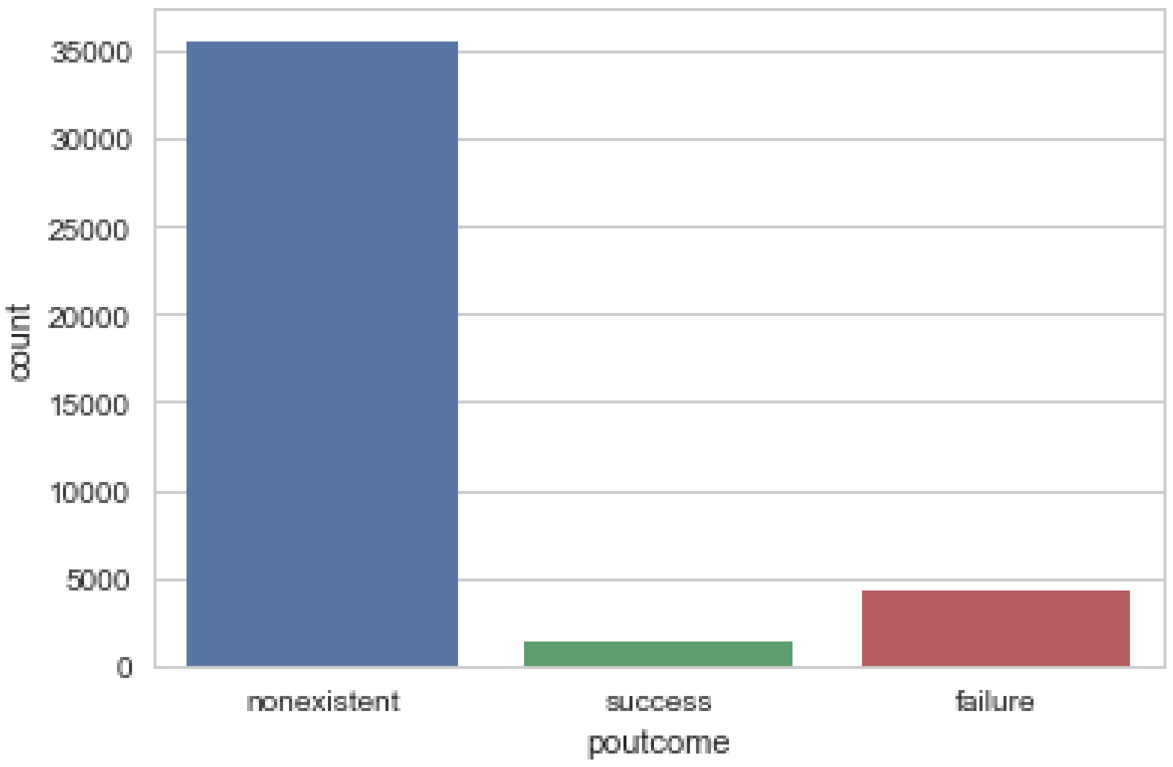
Barplot for personal loan

```
sns.countplot(x="loan", data=data)

plt.show()
```



Barplot for previous marketing campaign outcome

```
sns.countplot(x="poutcome", data=data)

plt.show()
```



Feature Selection

Our prediction will be based on the customer’s job, marital status, whether he(she) has credit in default, whether he(she) has a housing loan, whether he(she) has a personal loan, and the outcome of the previous marketing campaigns. So, we will drop the variables that we do not need.

```
data.drop(data.columns[[0, 3, 7, 8, 9, 10, 11, 12, 13, 15, 16,
17, 18, 19]], axis=1, inplace=True)
```

Data Preprocessing – Create dummy variables

In logistic regression models, encoding all of the independent variables as dummy variables allows easy interpretation and calculation of the odds ratios, and increases the stability and significance of the coefficients

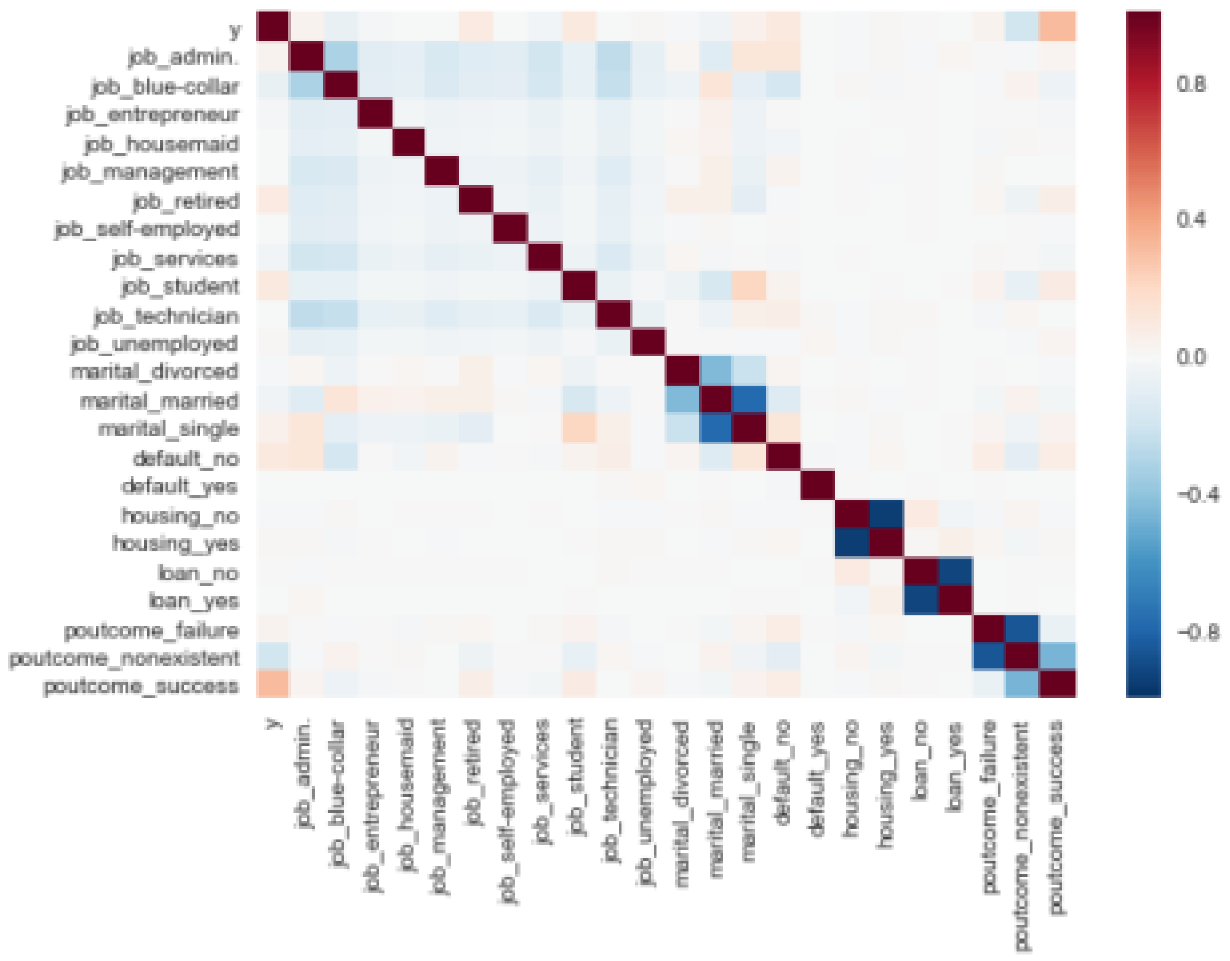
```
data2 = pd.get_dummies(data, columns =['job', 'marital',
'default', 'housing', 'loan', 'poutcome'])
```

Data Preprocessing – drop unknown columns

```
data2.drop(data2.columns[[12, 16, 18, 21, 24]], axis=1,
inplace=True)

data2.columns
```

Data Preprocessing – Check independence between independent variables

```
sns.heatmap(data2.corr())

plt.show()
```



Train test Split

```
X = data2.iloc[:,1:]

y = data2.iloc[:,0]

X_train, X_test, y_train, y_test = train_test_split(X, y,
random_state=0)
```

Fit Logistic Regression Model

```
classifier = LogisticRegression(random_state=0)

classifier.fit(X_train, y_train)
```

Predicting the test set results and creating confusion matrix

```
y_pred = classifier.predict(X_test)

from sklearn.metrics import confusion_matrix

confusion_matrix = confusion_matrix(y_test, y_pred)

print(confusion_matrix)
```

Accuracy

```
print('Accuracy of logistic regression classifier on test set:
{:.2f}'.format(classifier.score(X_test, y_test)))
```

Classification Report

```
from sklearn.metrics import classification_report

print(classification_report(y_test, y_pred))
```

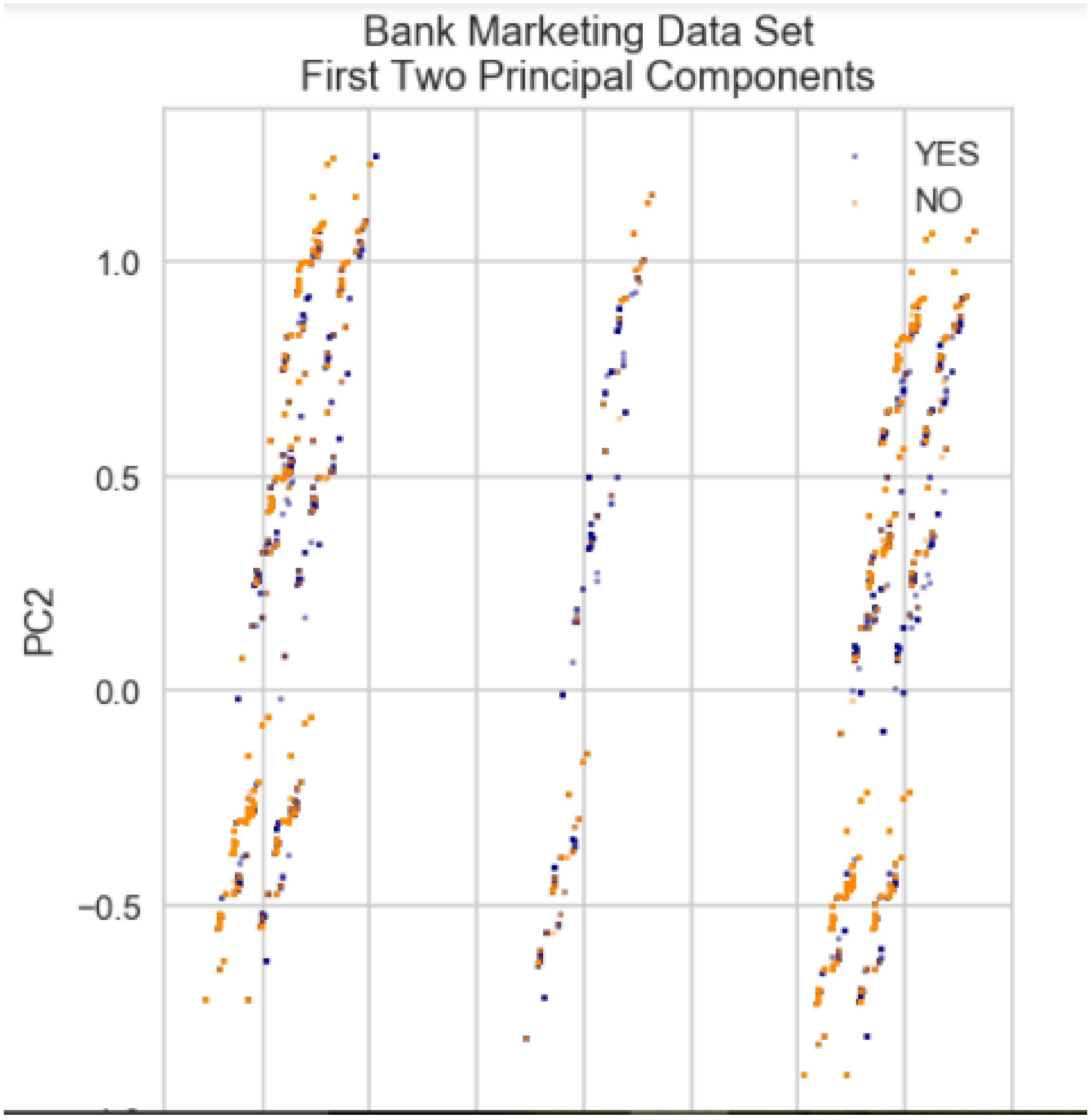
Interpretation

Of the entire test set, 88% of the promoted term deposit were the term deposit that the customers liked. Of the entire test set, 90% of the customer’s preferred term deposits that were promoted.

Classifier visualization playground

The purpose of this section is to visualize logistic regression classsifiers' decision boundaries. In order to better vizualize the decision boundaries, we'll perform Principal Component Analysis (PCA) on the data to reduce the dimensionality to 2 dimensions

```
from sklearn.decomposition import PCA

X = data2.iloc[:,1:]

y = data2.iloc[:,0]

pca = PCA(n_components=2).fit_transform(X)

X_train, X_test, y_train, y_test = train_test_split(pca, y,
random_state=0)

plt.figure(dpi=120)

plt.scatter(pca[y.values==0,0], pca[y.values==0,1],
alpha=0.5, label='YES', s=2, color='navy')

plt.scatter(pca[y.values==1,0], pca[y.values==1,1],
alpha=0.5, label='NO', s=2, color='darkorange')

plt.legend()

plt.title('Bank Marketing Data Set\nFirst Two Principal
Components')

plt.xlabel('PC1')

plt.ylabel('PC2')

plt.gca().set_aspect('equal')

plt.show()
```


Bank Marketing Data Set
First Two Principal Components
PC2
1.0
0.5
0.0
−0.5
YES
NO