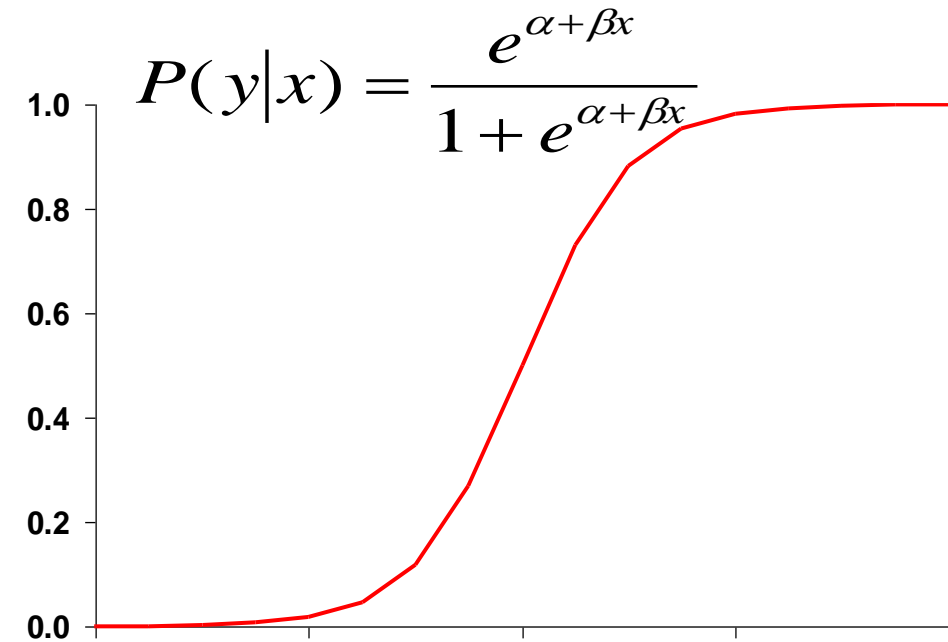




LOGISTIC REGRESSION

Anshu Pandey

- Logistic Regression is a 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.



WHAT IS LOGISTIC REGRESSION?

Logistic Regression 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

- * 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.
- * The independent variables should be independent of each other. That is, the model should have little or no multicollinearity.
- * The independent variables are linearly related to the log odds.
- * Logistic regression requires quite large sample sizes.

Logistic Regression – Diabetes Dataset

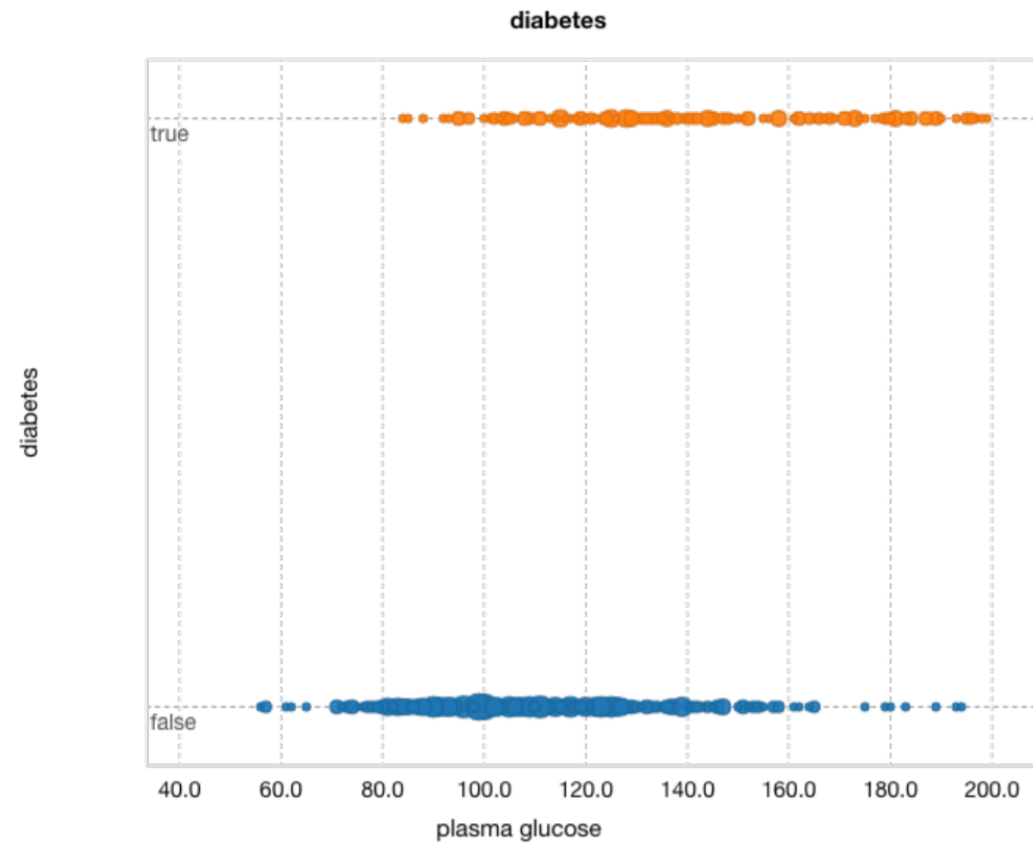
Y axis

True: Person has diabetes

False: No diabetes

X Axis

Feature – plasma glucose

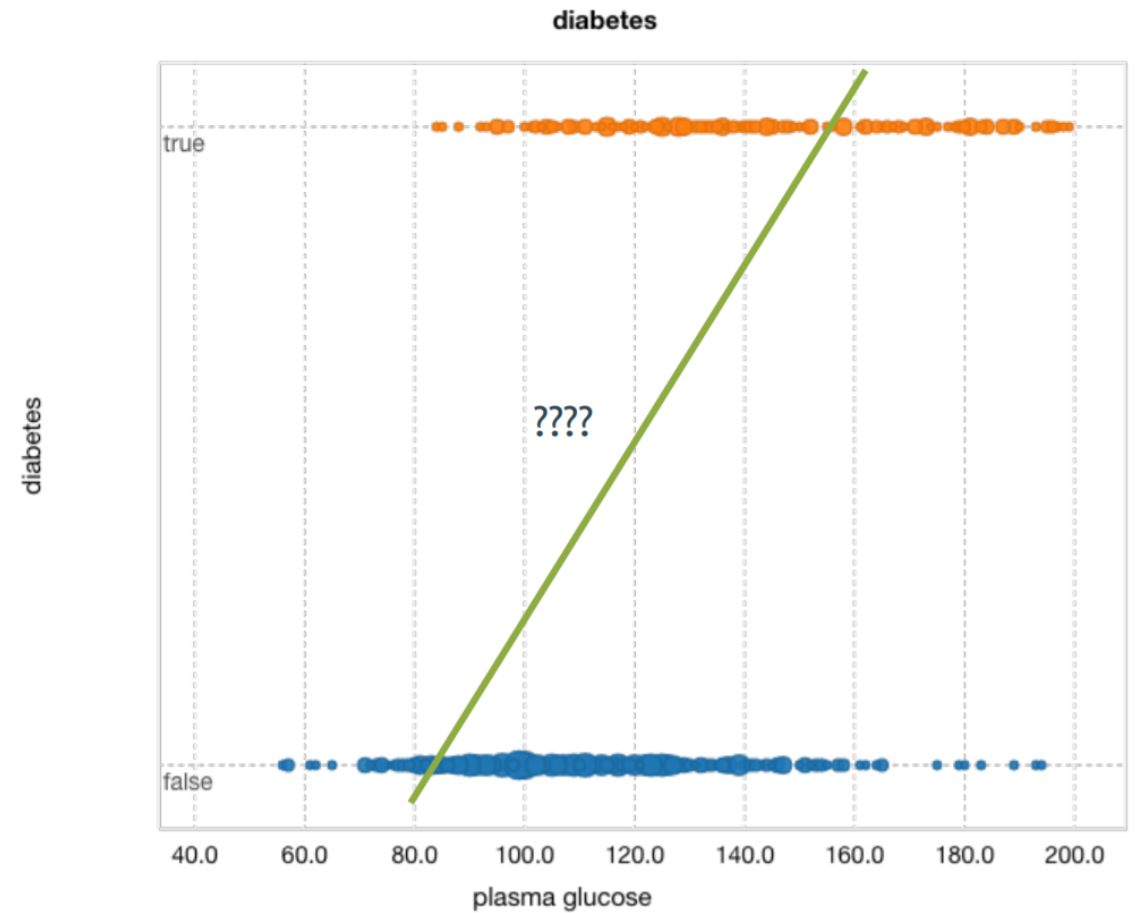


Logistic Regression – Diabetes Dataset

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

\hat{y} = Value predicted by current Algorithm

Linear Regression in one Variable

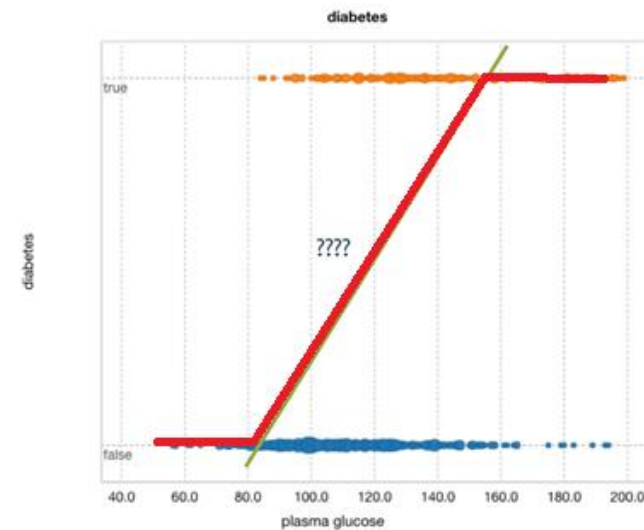
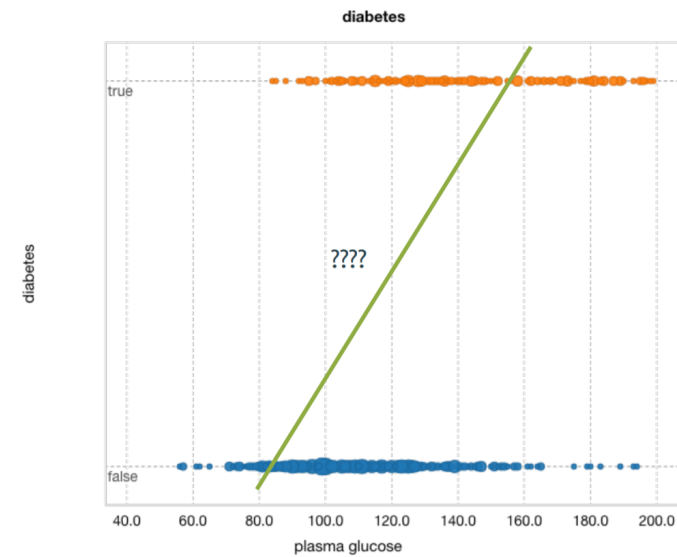


Logistic Regression – Diabetes Dataset

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



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

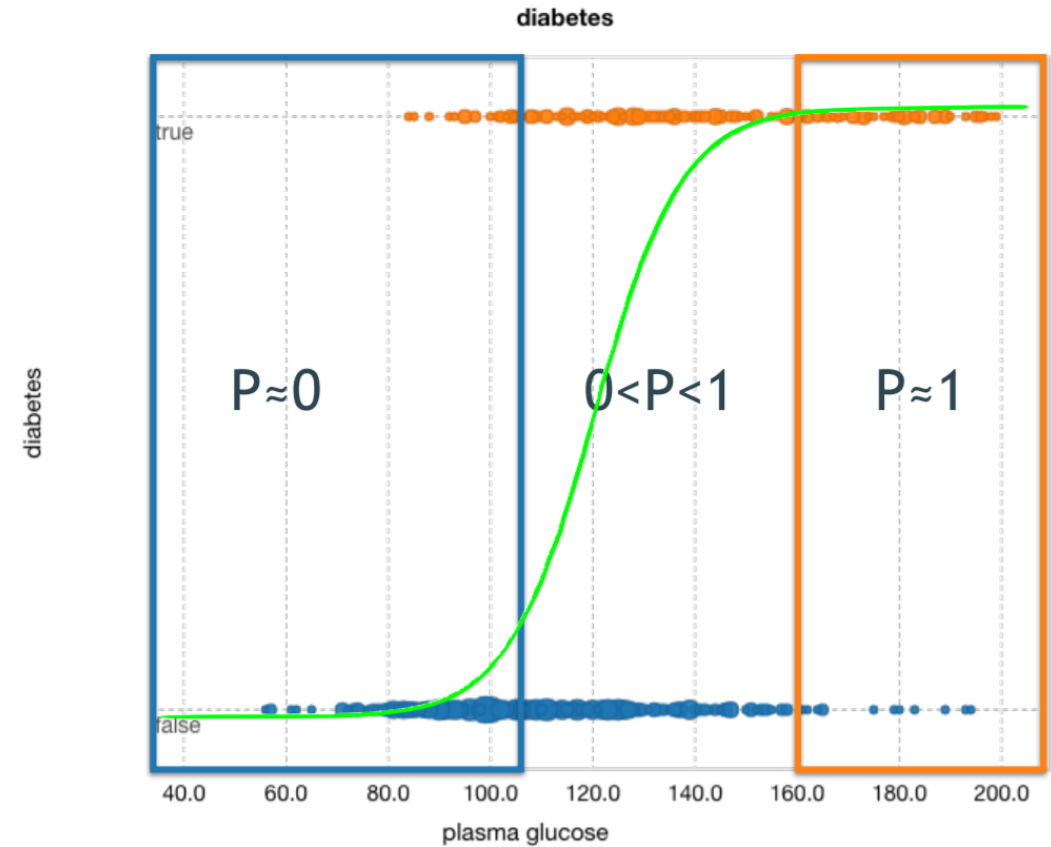


Logistic Regression – Diabetes Dataset

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



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

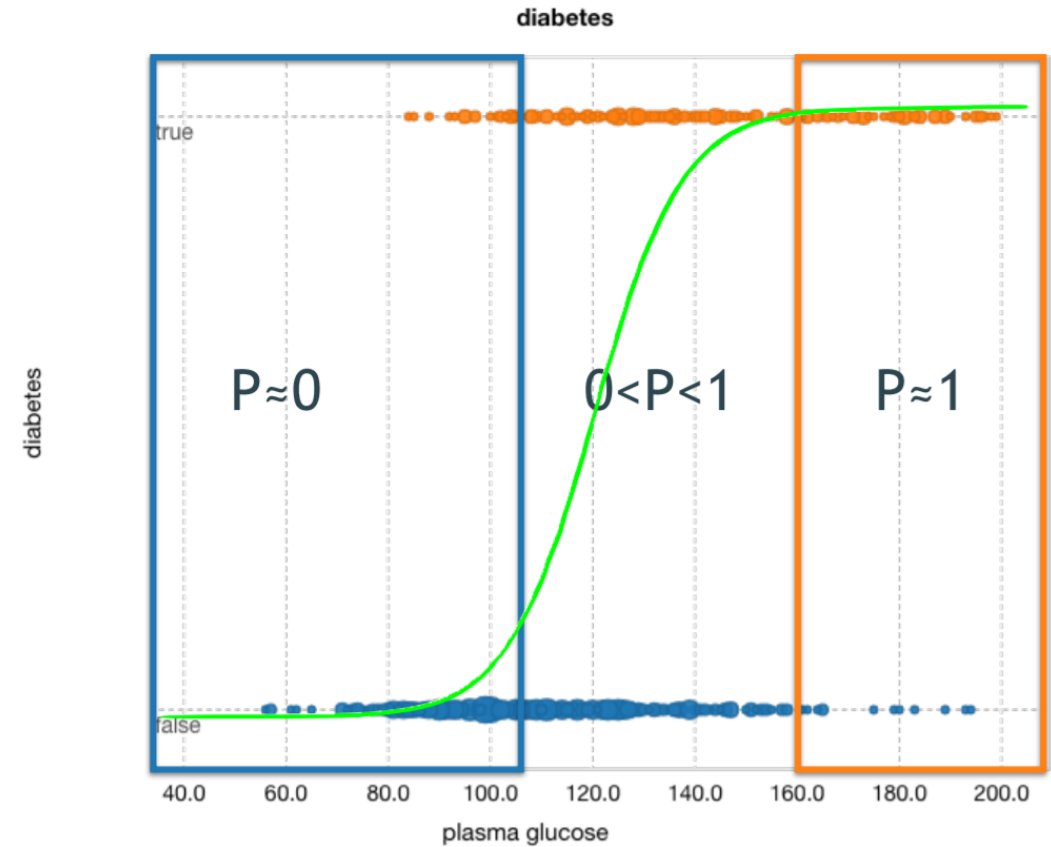


Logistic Regression – Diabetes Dataset

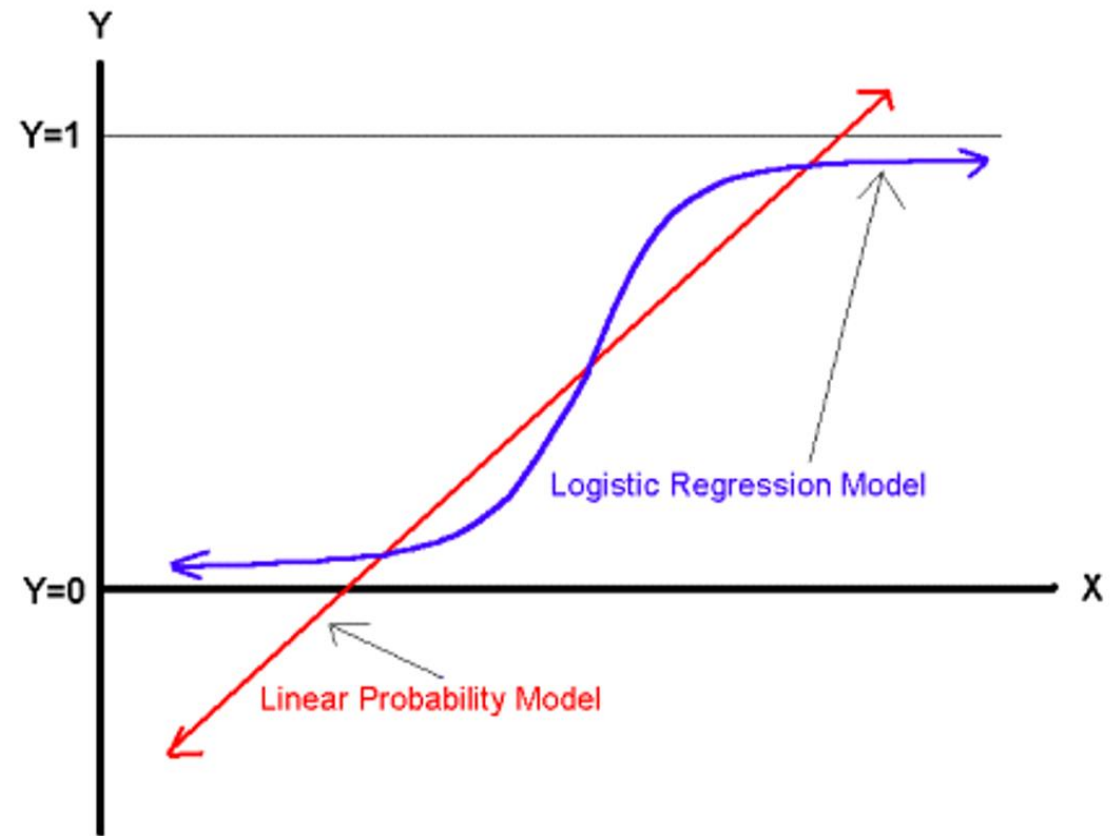
$$p = \frac{1}{1 + e^{-(mx+c)}}$$



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



Comparing Linear Probability Model and Logistic Regression Model



Binary Classification using Logistic Regression

Sigmoidal Function

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

Multinomial Classification using Logistic Regression

Softmax function

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

Maximum Likelihood Estimation (MLE)

MLE is a statistical method for estimating the coefficients of a model.

The likelihood function (L) measures the probability of observing the particular set of dependent variable values (p_1, p_2, \dots, p_n) that occur in the sample:

$$L = \text{Prob} (p_1 * p_2 * \dots * p_n)$$

The higher the L , the higher the probability of observing the p s in the sample.

Maximum Likelihood Estimation (MLE)

MLE involves finding the coefficients (β_0, β_1) that makes the log of the likelihood function ($LL < 0$) as large as possible

Or, finds the coefficients that make -2 times the log of the likelihood function ($-2LL$) as small as possible

The maximum likelihood estimates solve the following condition:

$$\{Y - p(Y=1)\}X_i = 0$$

summed over all observations, $i = 1, \dots, n$

Objective of Logistic Regression

Binary Classification:

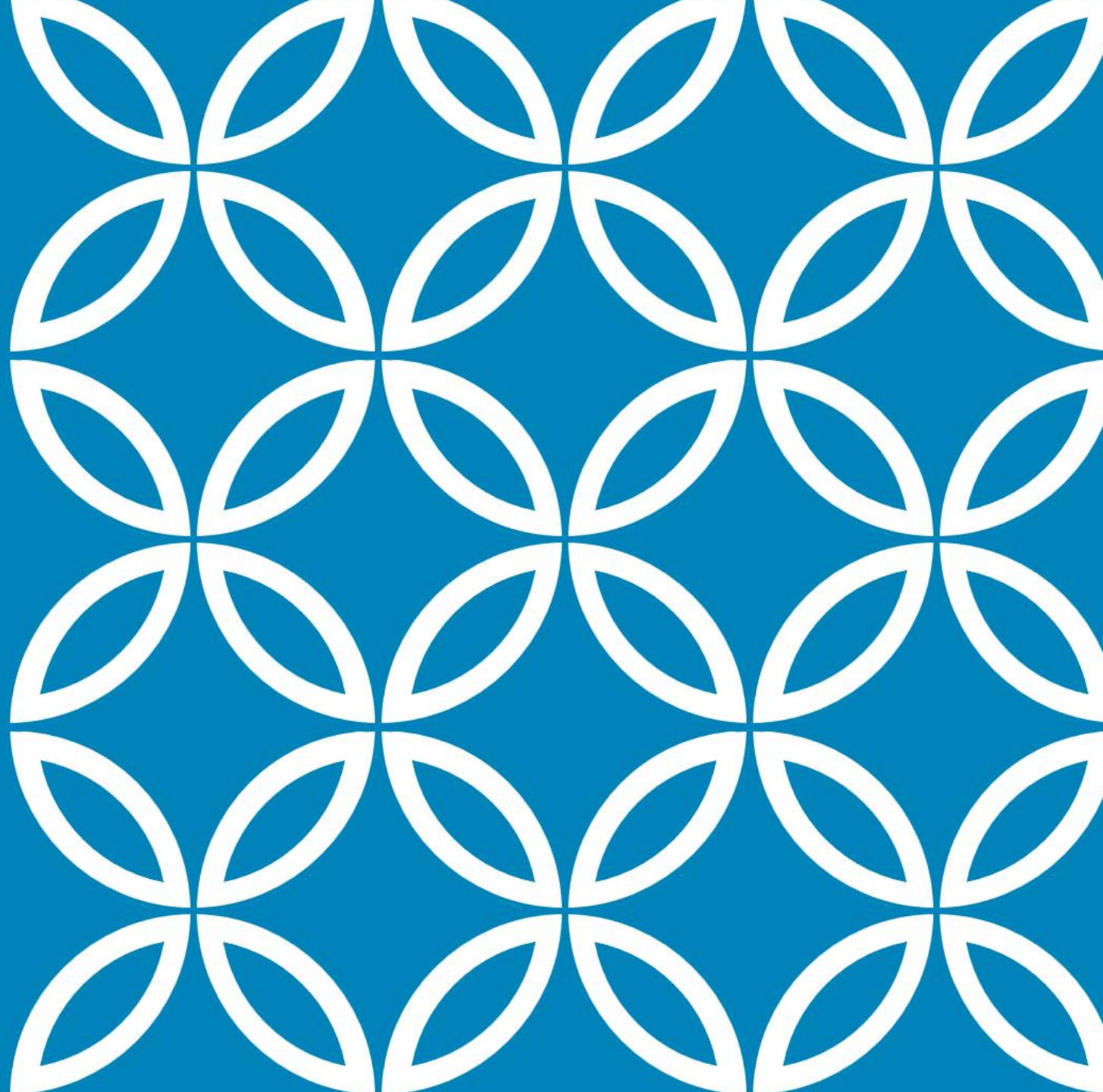
- Given the subject and the email text predicting, Email Spam or not.
- Sunny or rainy day prediction, using the weather information.
- Based on the bank customer history, Predicting whether to give the loan or not.

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

MODEL EVALUATION

Anshu Pandey



Confusion Matrix

	Positive	Negative
Positive	True Positive	False Positive
Negative	False Negative	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

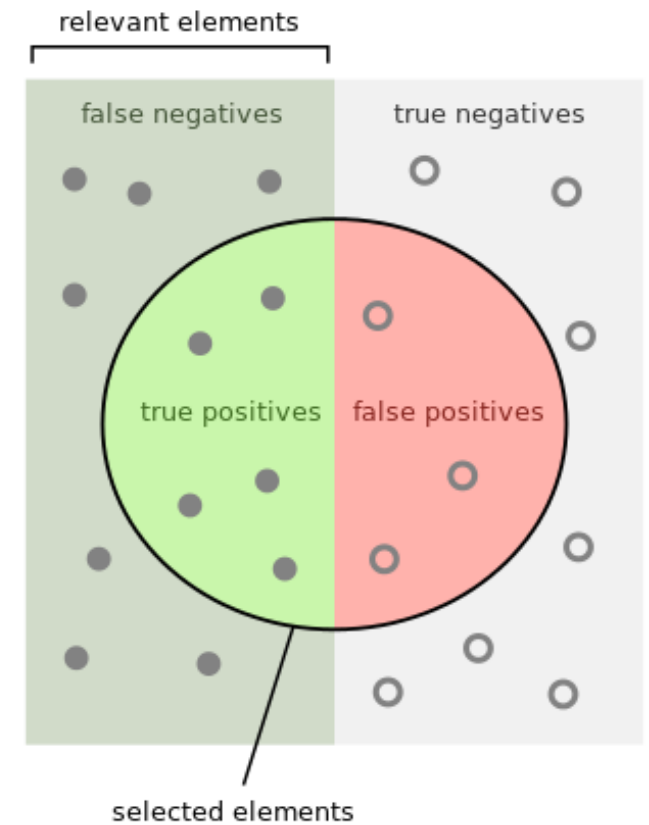
Accuracy

$$\text{Accuracy} = \frac{tp + tn}{tp + tn + fp + fn}$$

Relevancy Score

1. PRECISION

2. RECALL



How many selected items are relevant?

$$\text{Precision} = \frac{\text{true positives}}{\text{true positives} + \text{false positives}}$$

How many relevant items are selected?

$$\text{Recall} = \frac{\text{true positives}}{\text{true positives} + \text{false negatives}}$$

Recall

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.

$$\text{Recall} = \frac{tp}{tp + fn}$$

Precision

The fraction of relevant instances among the retrieved instances.

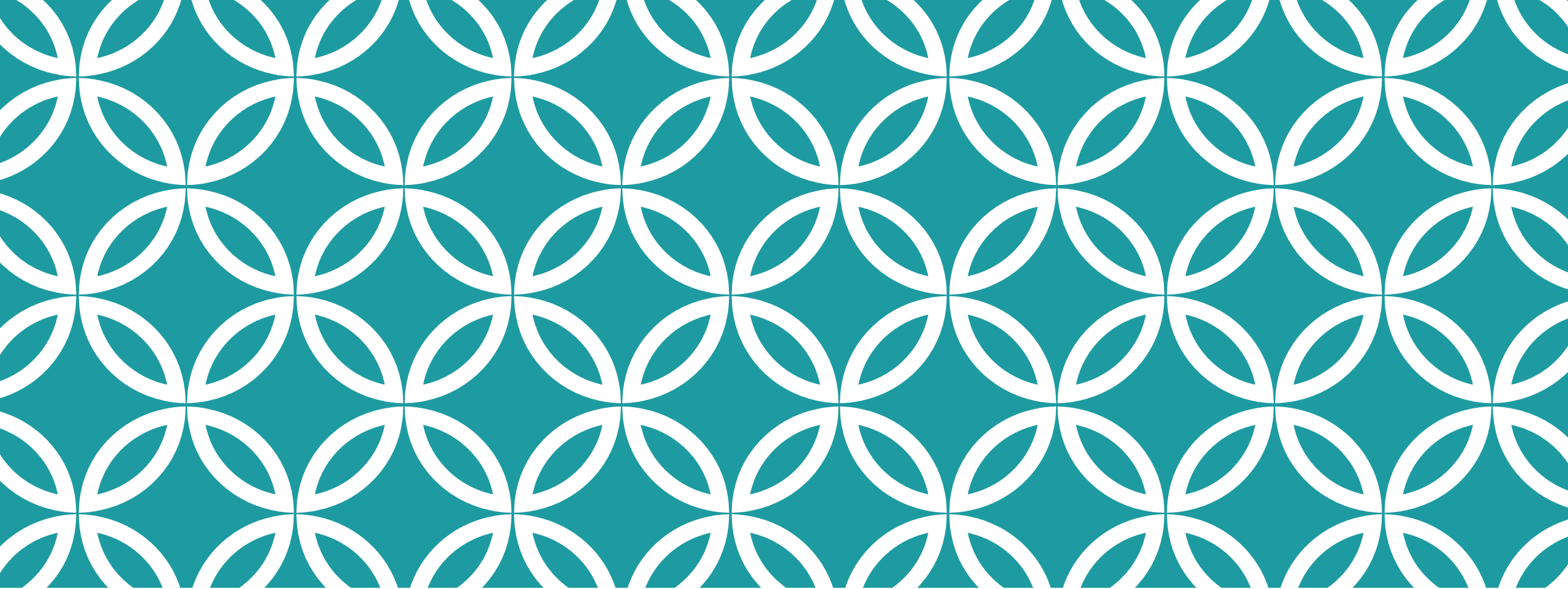
The precision is intuitively the ability of the classifier to not label a sample as positive if it is negative.

$$\text{Precision} = \frac{tp}{tp + fp}$$

F1 Score

A measure that combines precision and recall is the harmonic mean of precision and recall.

$$F^1 = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}$$



Logistic Regression

Python code using sklearn

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)
```

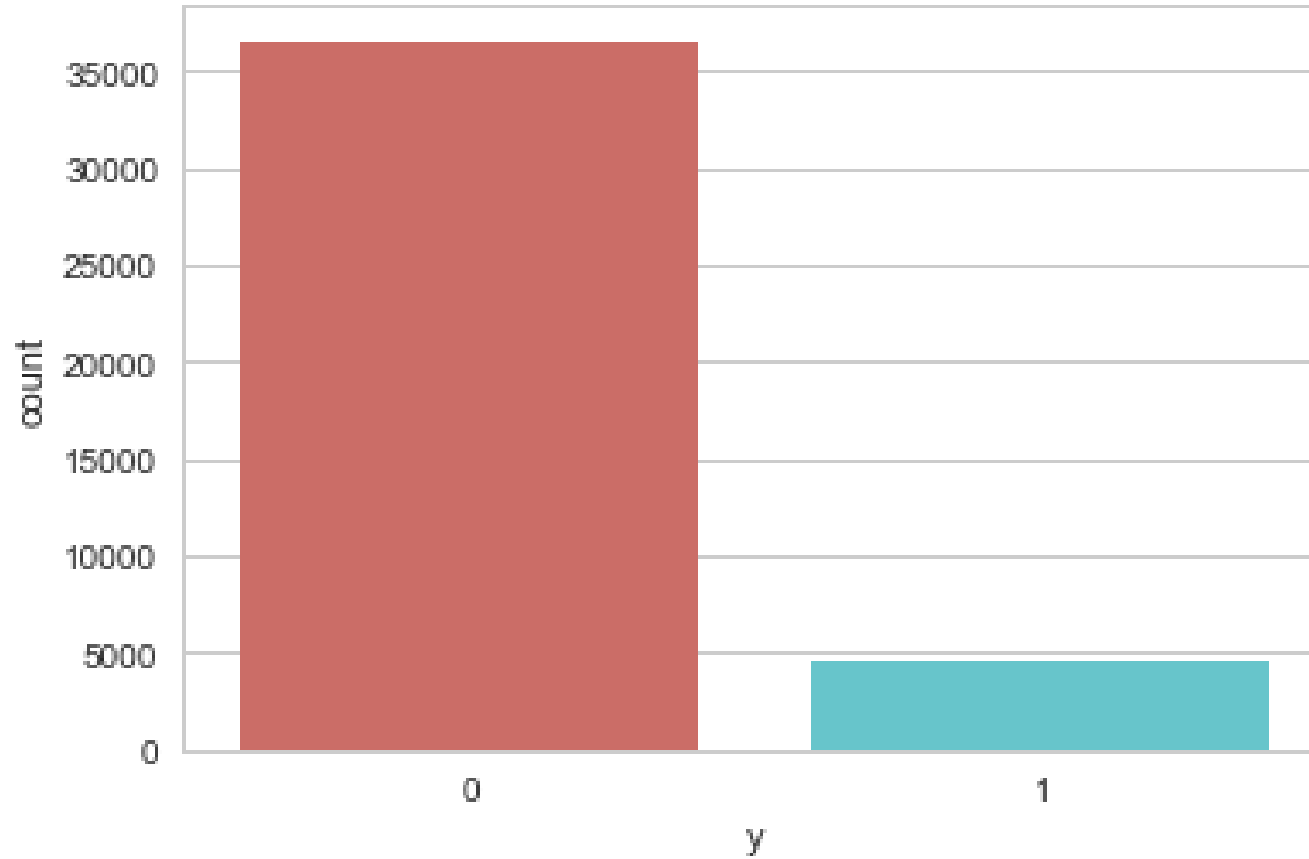
Import dataset

```
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()
```

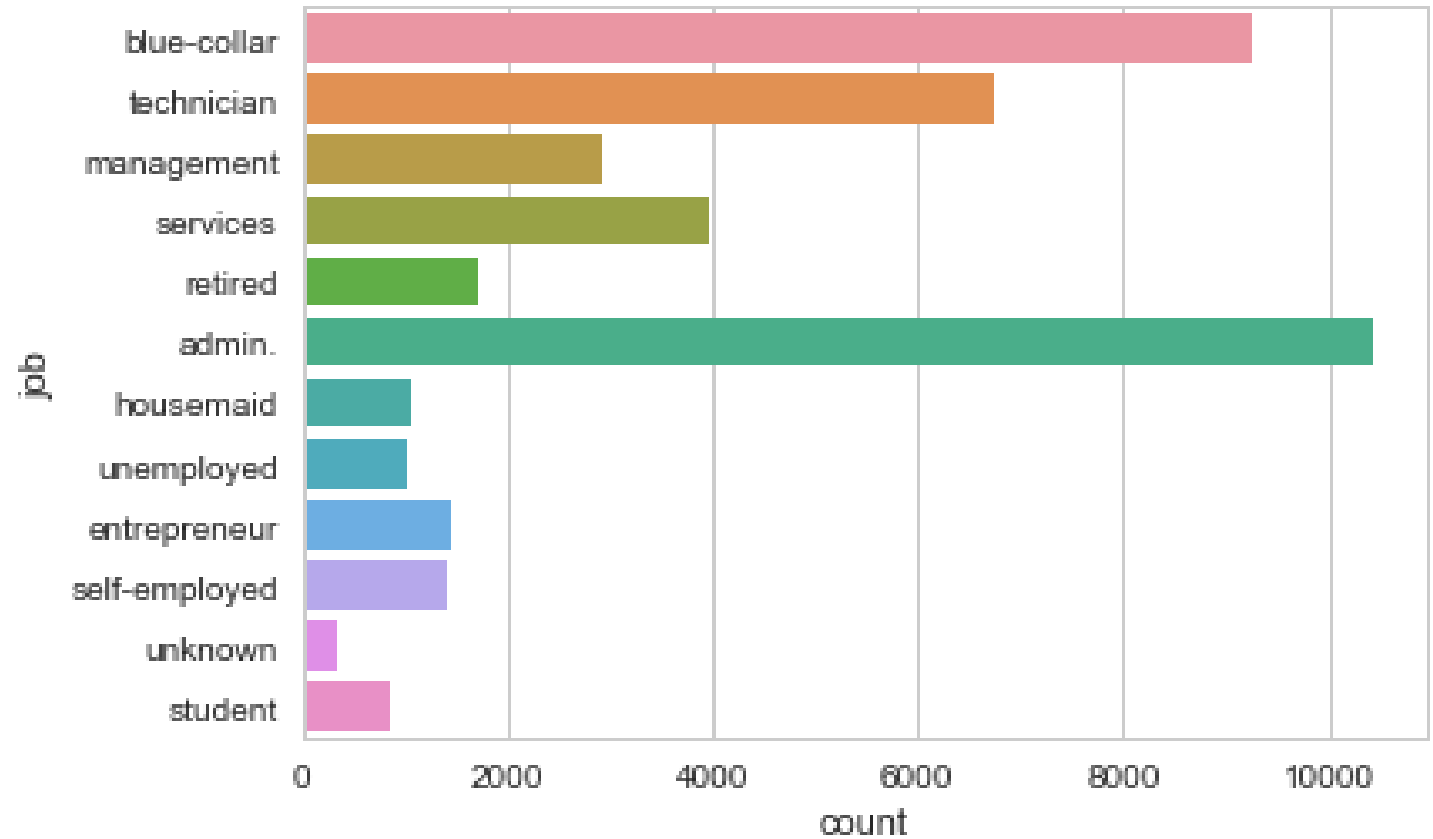
Barplot for dependent Variable



Customer job distribution

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

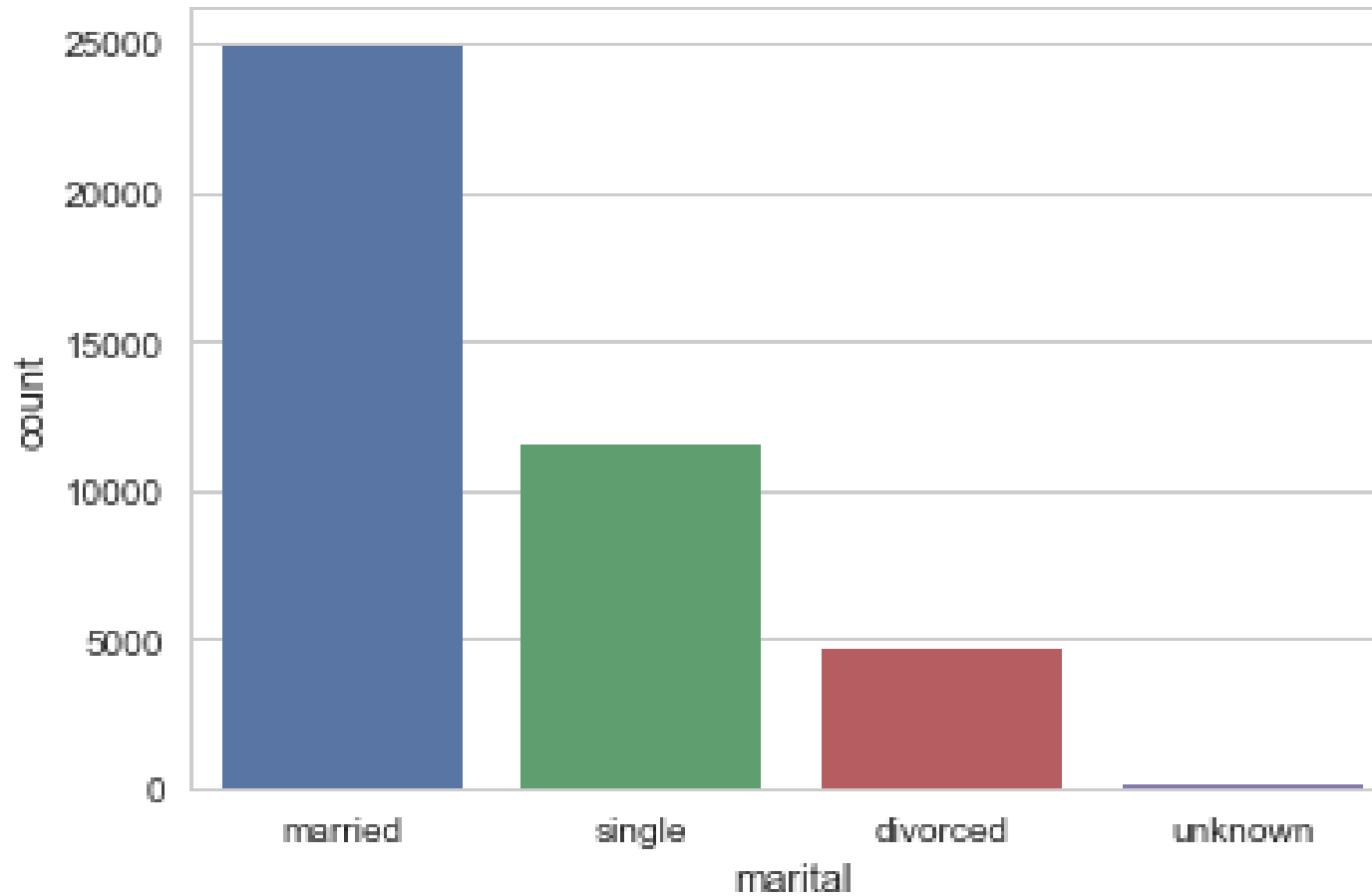
Customer job distribution



Customer Marital Status Distribution

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

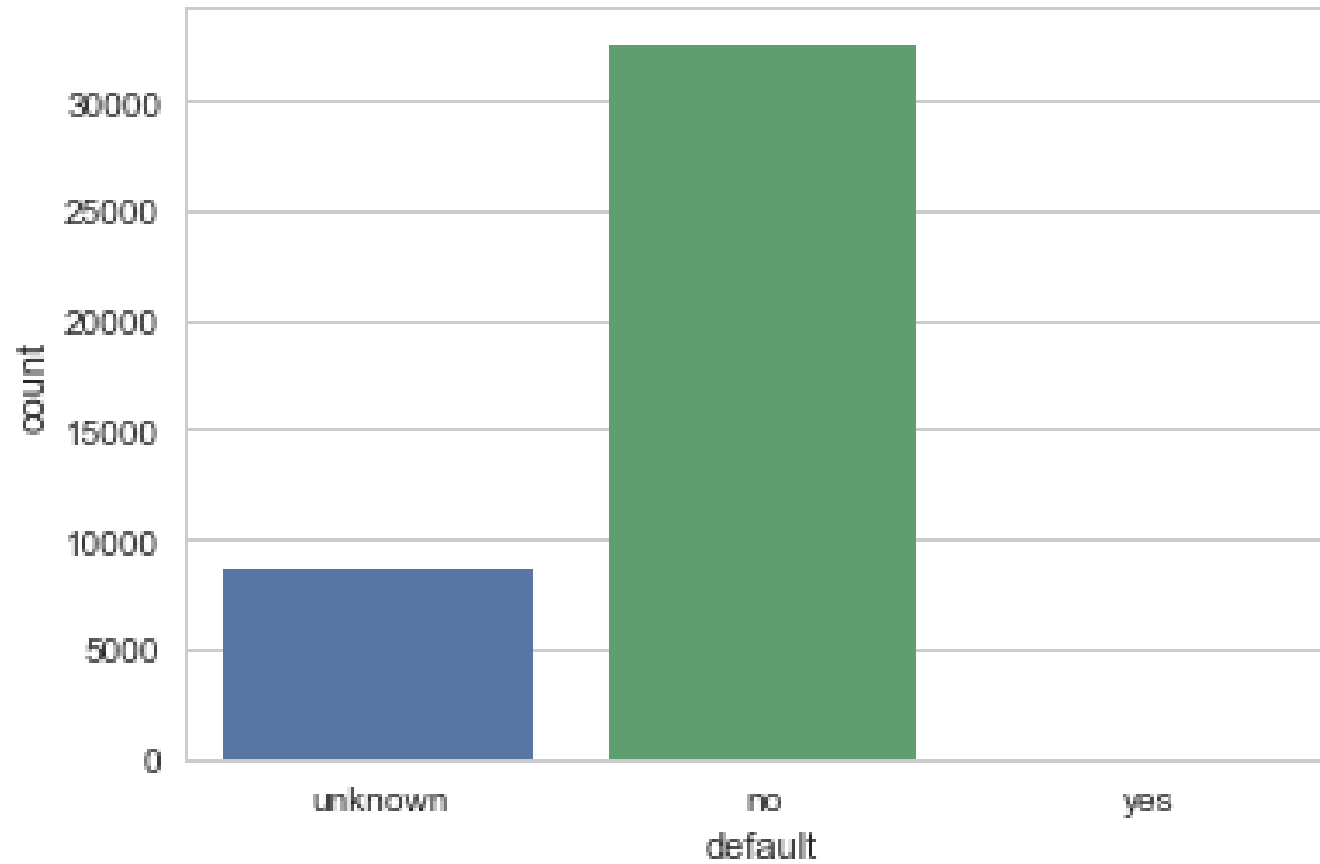
Customer Marital Status Distribution



Barplot for Credit in default

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

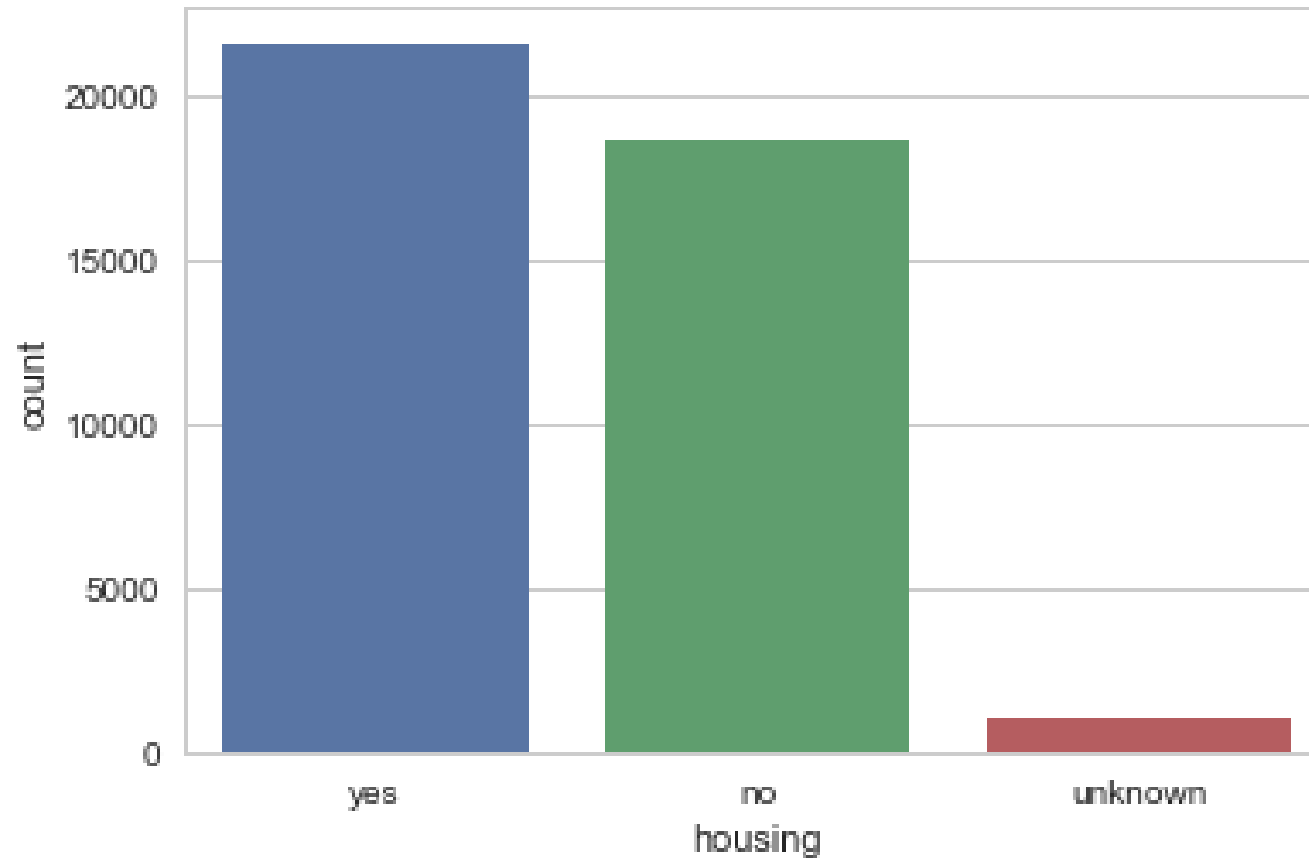
Barplot for Credit in default



Barplot for housing loan

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

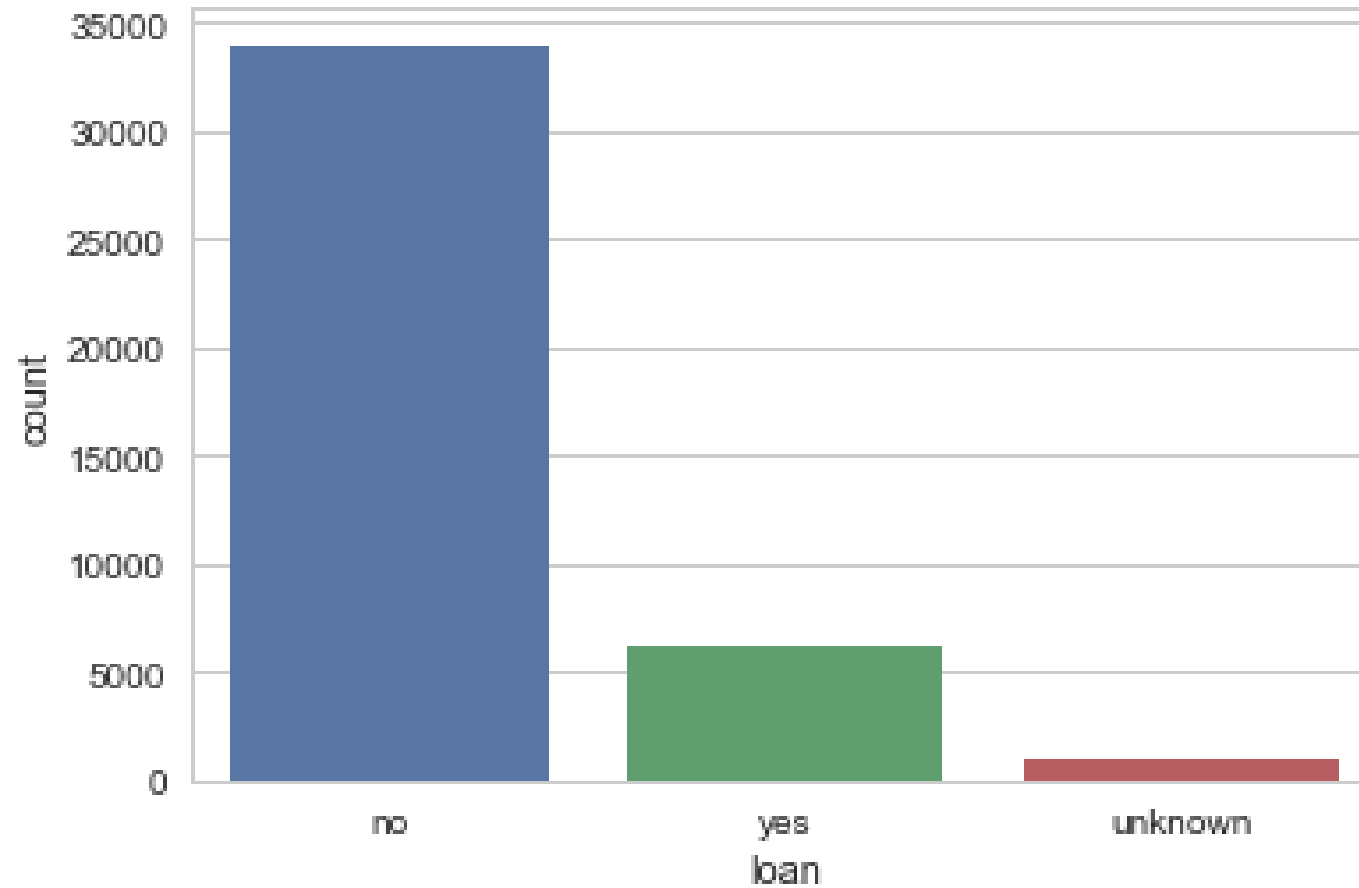
Barplot for housing loan



Barplot for personal loan

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

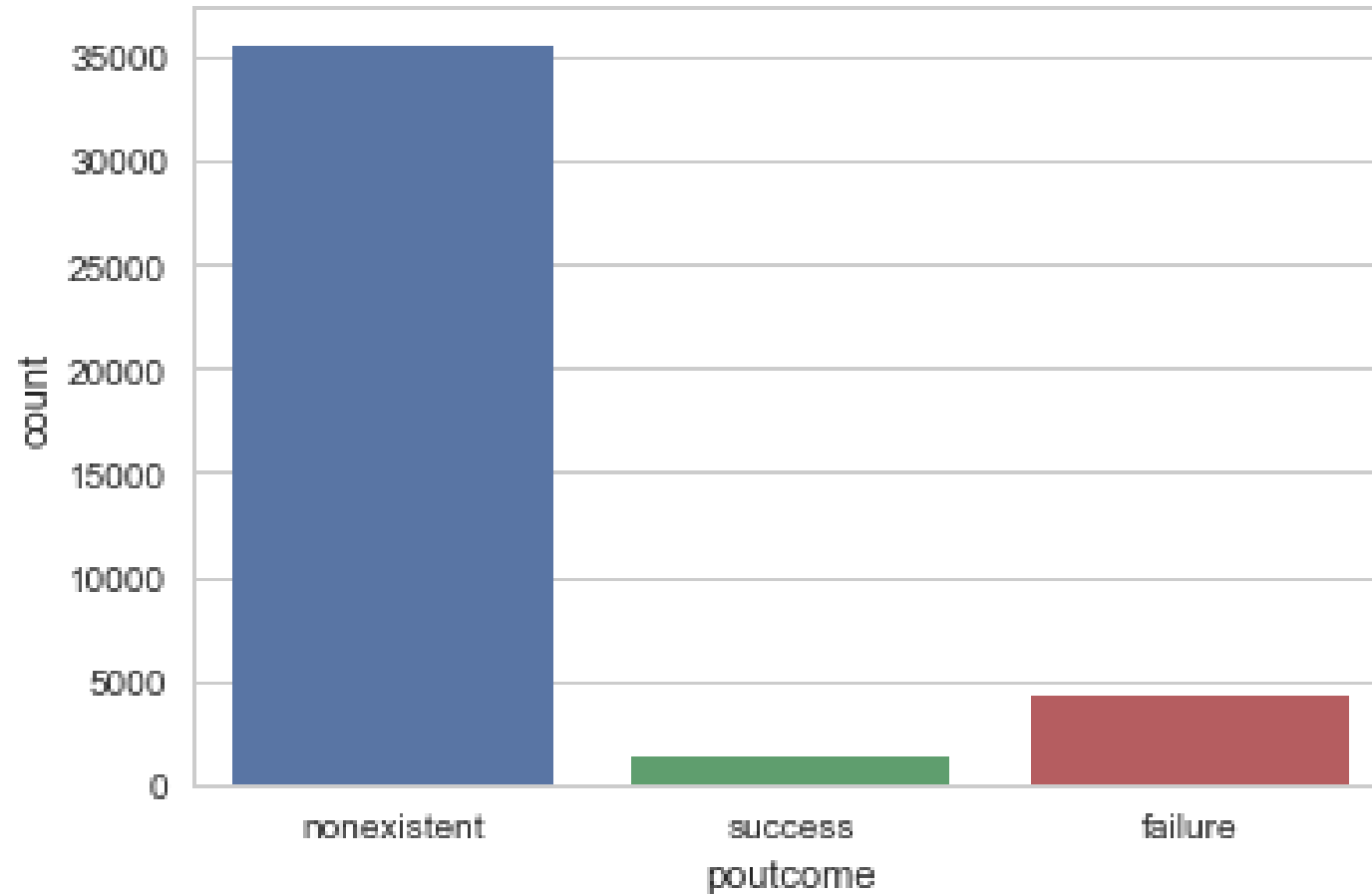
Barplot for personal loan



Barplot for previous marketing campaign outcome

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

Barplot for previous marketing campaign outcome



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

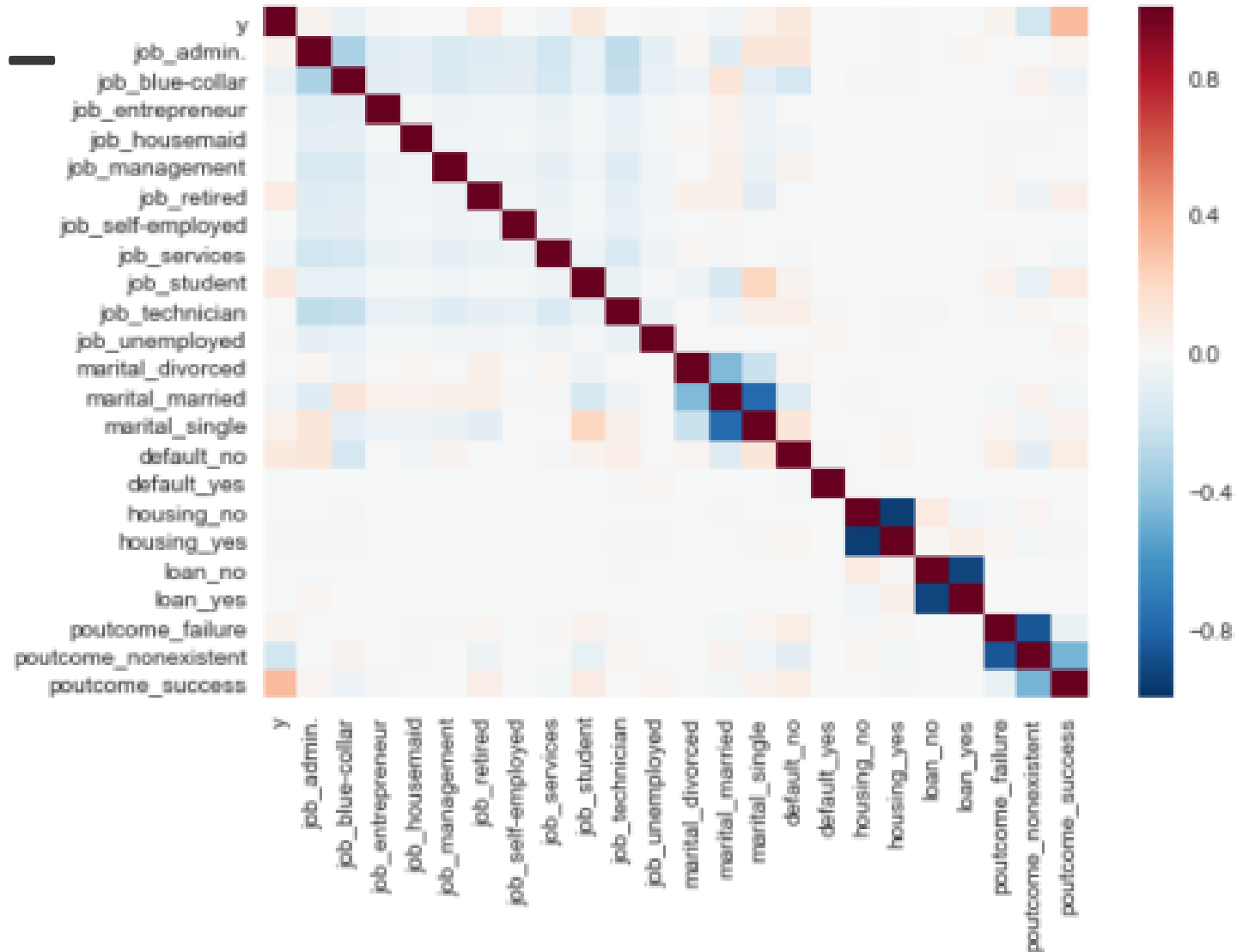
```
data2.drop(data2.columns[[12, 16, 18, 21, 24]], axis=1,  
inplace=True)  
data2.columns
```

Data Preprocessing – Check independence between independent variables

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

Data Preprocessing –

Check independence
between independent
variables



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 classifiers' decision boundaries. In order to better visualize 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)
```

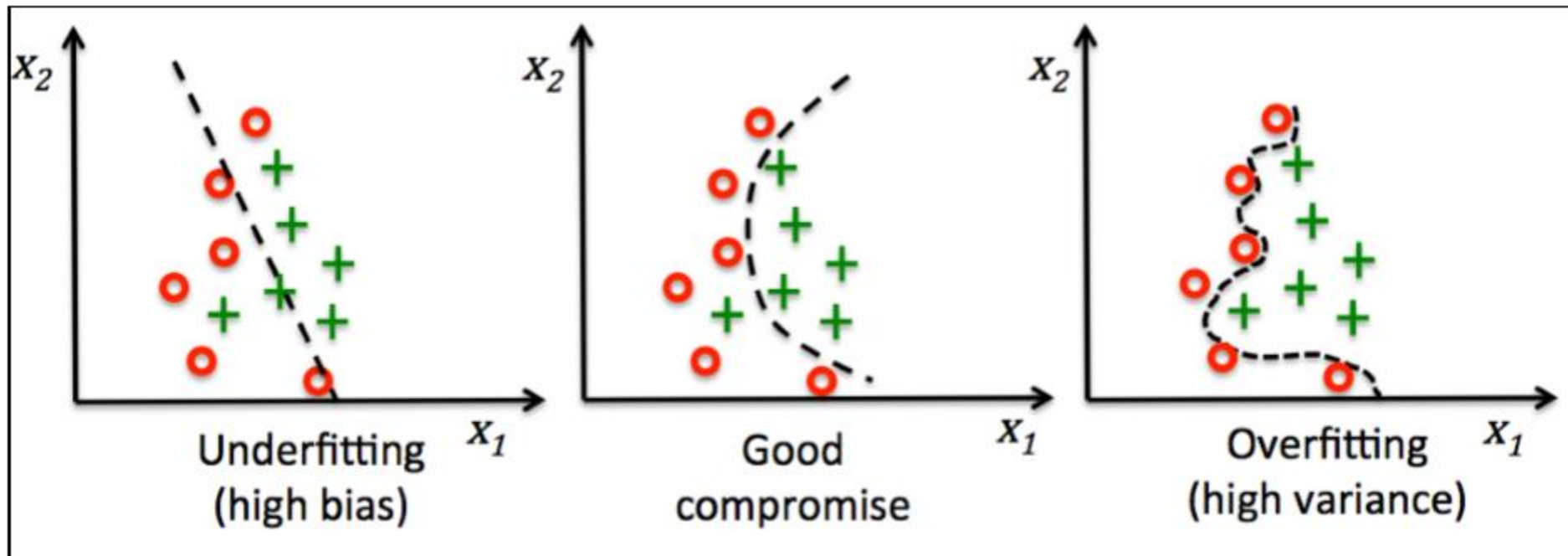
Classifier visualization playground

```
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()
```

Classifier visualization playground

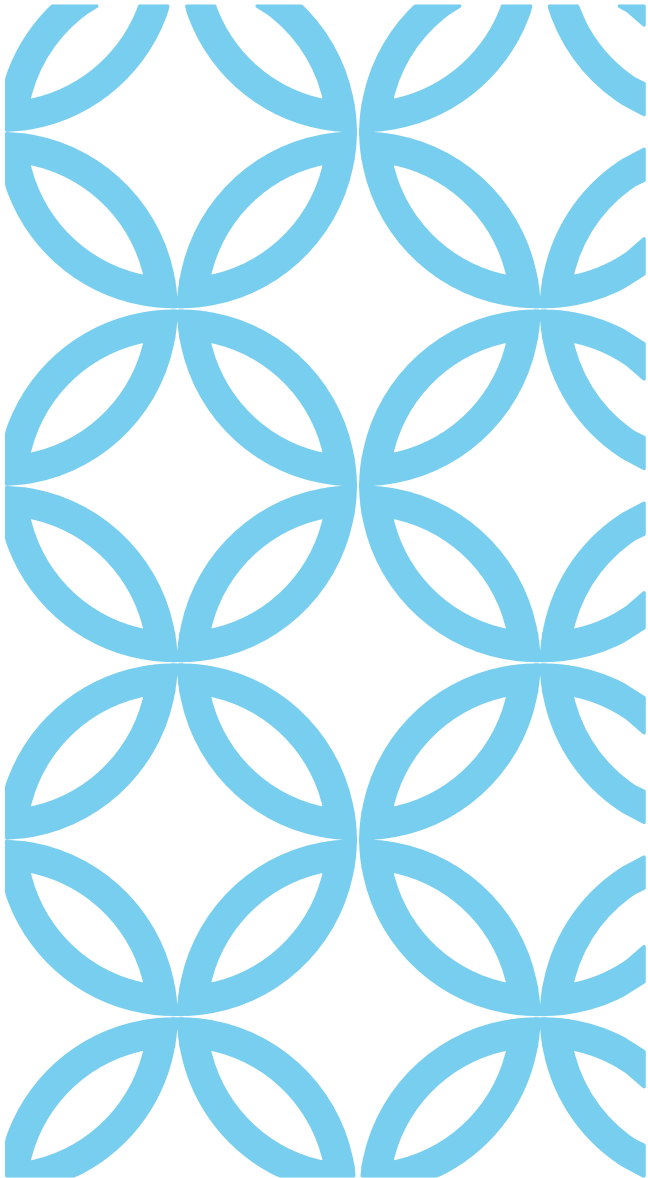


Overfitting & Generalisation



Regularization for logistic Regression

```
sklearn.linear_model.LogisticRegression(penalty='l2', dual=False,  
tol=0.0001, C=1.0, fit_intercept=True, intercept_scaling=1,  
class_weight=None, random_state=None, solver='liblinear',  
max_iter=100, multi_class='ovr', verbose=0, warm_start=False,  
n_jobs=1
```



Stay tuned for practicing Logistic Regression with datasets

THANK YOU