Predictive Analytics for a Bank Marketing Campaign

Feature Selection Report

Robert Maheux

March 4 2019

Contents

[1. Introduction 3](#_Toc3900584)

[2. Separate the Dataset Features into X and Y Datasets 4](#_Toc3900585)

[3. Perform Primary Component Analysis on the Dependent Variables 5](#_Toc3900586)

[4. Estimate the Importance of Features using Extra Trees Classifier Decision Tree 8](#_Toc3900587)

[5. Remove the Least Important Features 11](#_Toc3900588)

[6. Split the Dataset into Training and Testing Sets 12](#_Toc3900589)

[7. Summary 14](#_Toc3900590)

[Appendix A – Listing of Python Commands Used for this Report 15](#_Toc3900591)

# 1. Introduction

This report describes the steps taken to reduce the number of features in the bank marketing dataset.

The *Primary Component Analysis* technique will be applied to the dataset. If the results are found to have limited value, an *Extra Trees Classifier* decision tree will be used. The features that are found to have minimal importance will be removed from the dataset. Finally, the data will be split into training and testing sets.

After these steps have been completed, the dataset will be ready to be used by the model building process.

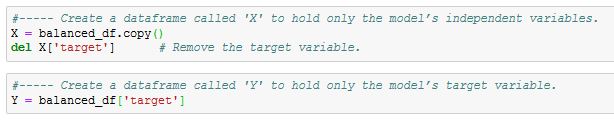
# 2. Separate the Dataset Features into X and Y Datasets

Split the dataset into two separate data frames.

The first data frame (called ‘X’) will contain only the independent variables. It will be used for the feature selection process.

The second data frame (called ‘Y’) will contain only the target variable.

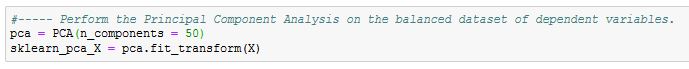
Both of these data frames will be used to build predictive models.



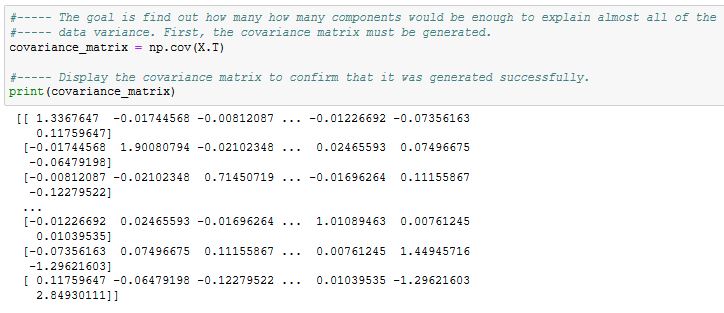
# 3. Perform Primary Component Analysis on the Dependent Variables

This section describes the steps to perform the *Primary Component Analysis* (PCA). The results will indicate whether the list of 50 dependent variables can be reduced to a smaller number of components.

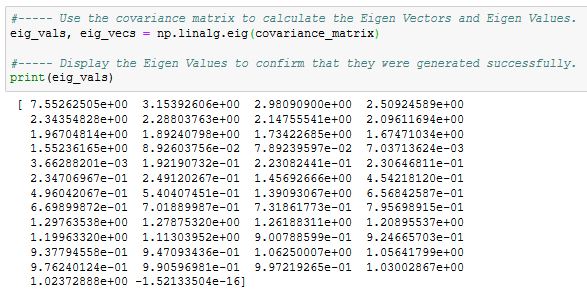
Use sklearn’s *PCA* function on the balanced dataset of dependent variables:



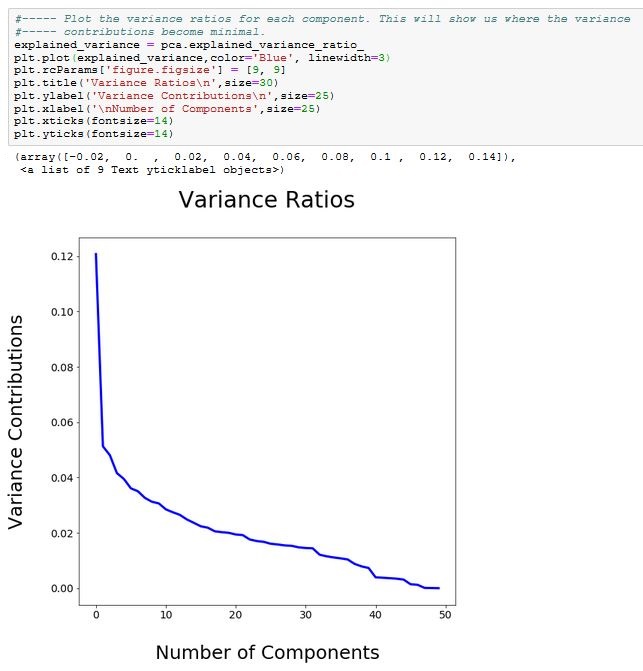
The goal is to find out how many components would be enough to explain almost all of the data variance. First, calculate the covariance matrix:



Use the covariance matrix to calculate the Eigen vectors and Eigen values:

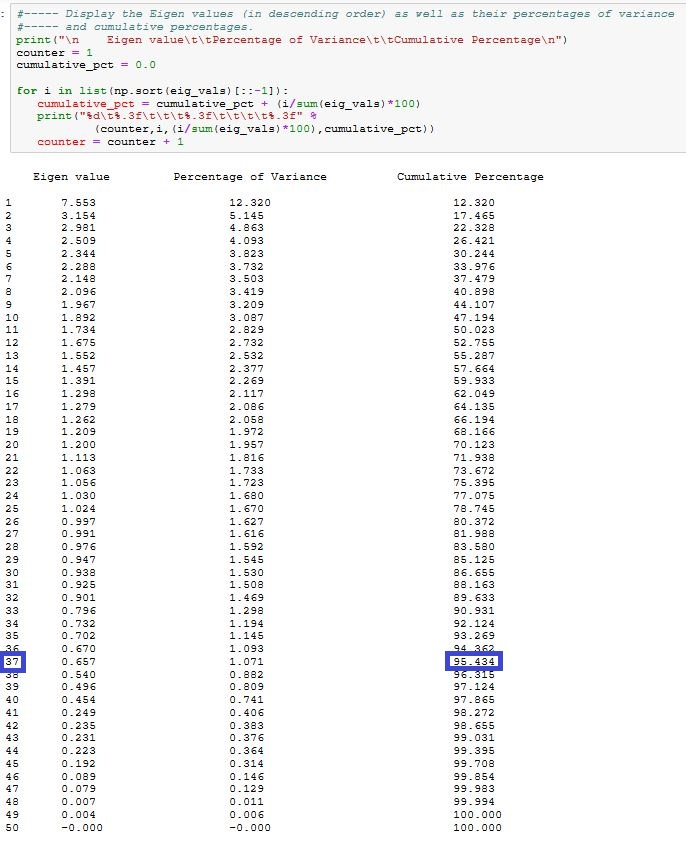


Plot the variance ratios for each component in order to visually see the reduction rate of the variances:



The following figure displays the Eigen values (in descending order) as well as their percentages of variance. For instance, the first component explains 12.3% of the variance, the second component explains 5.1%, and so on.

The last column displays the cumulative percentage values. It can be used to determine how many components we should keep in order to explain almost all of the data variance. If we wish to retain the components that represent **95%** **of the total variance**, then we’ll need to retain **37 components**.



The primary component analysis has shown that it would take 37 components to explain at least 95% of the variance. This doesn’t represent a substantial reduction of features and so the benefit of using PCA on this dataset would be marginal at best.

Furthermore, the use of the PCA technique would cause the individual features to be replaced with generic components. This would eliminate the ability to interpret how the individual features contribute to the outcomes. This type of information may be valuable for similar marketing campaigns.

Since the use of the PCA technique on this dataset would have minimal benefits and would result in the loss of valuable interpretative information, it will not be used to reduce the number of features.

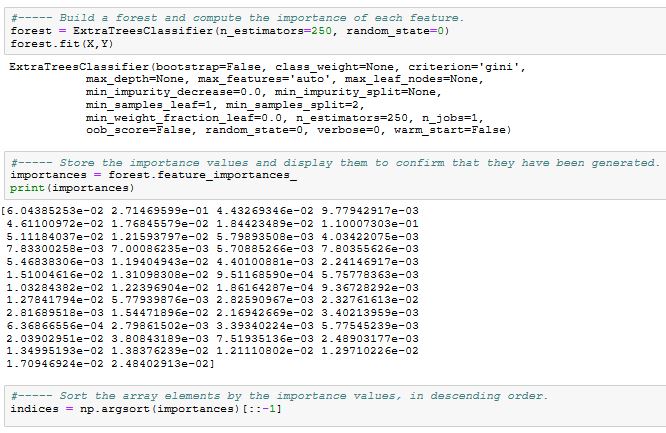
# 4. Estimate the Importance of Features using Extra Trees Classifier Decision Tree

The PCA technique offered little benefit for the bank marketing dataset but there are other feature reduction techniques that can be explored such as the *Extra Trees Classifier* decision tree.

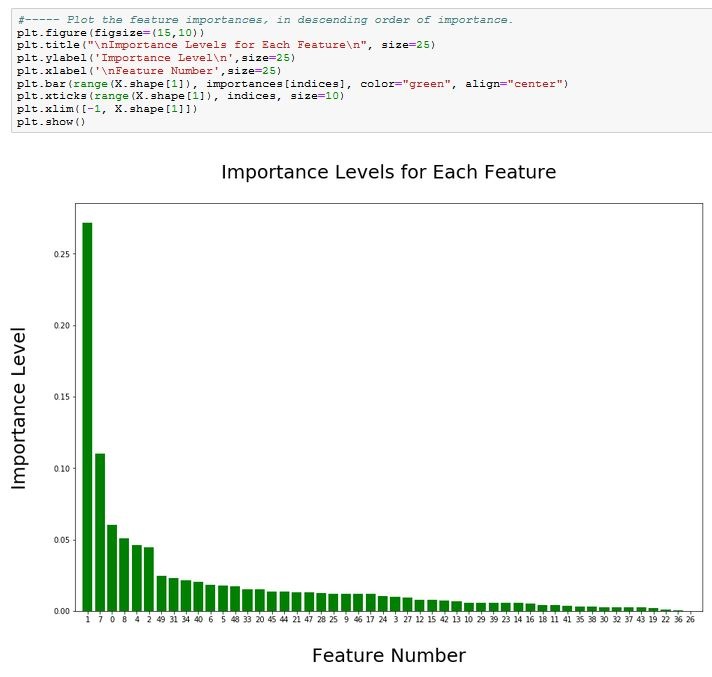
The term “Extra Trees” is short for **EXT**remely **RA**ndomized **TREE**s. It’s a variant of a random forest but at each step, the entire sample is used and splits are selected randomly rather than using specific criteria.

This technique can be used to compute the importance of each feature. The main benefit is that the individual features will not be replaced by generic components. It allows the least important features to be discarded while retaining the ability to interpret how the remaining features could impact the outcomes.

First, use sklearn’s *ExtraTreesClassifier* function to build a forest and compute the importance of each feature. Store the importance scores and sort them in descending order. The higher the score, the more important the feature is in terms of predicting the outcomes. The sum of all scores is 100%.



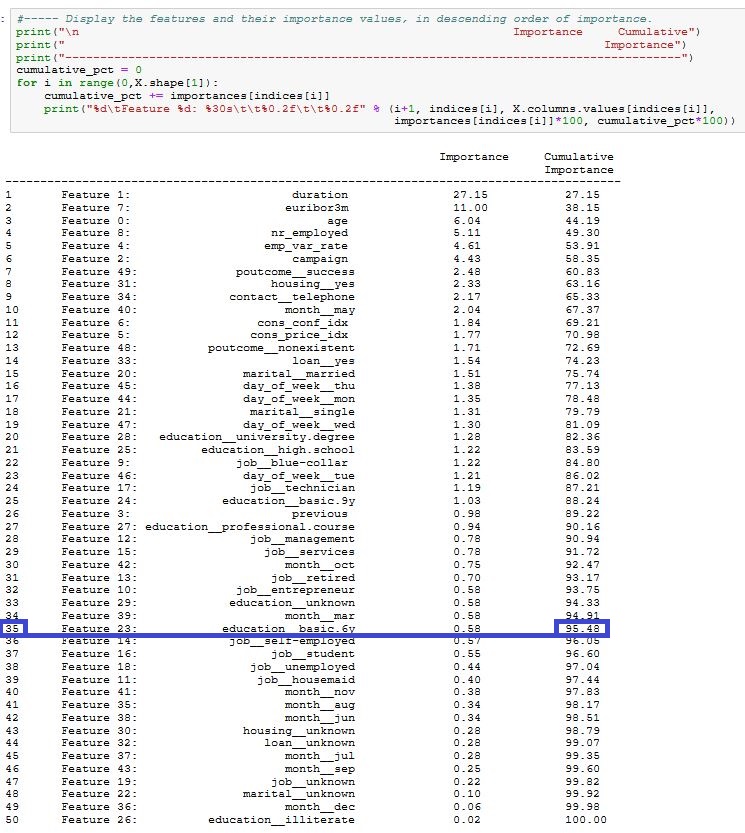
Plot the importance scores for each feature, in descending order of importance:



The bar chart illustrates that the first six features are much more important than the others, but a lot of important information would be lost if the remaining 44 features were discarded. The number of discarded features is subjective. In order to preserve as much information as possible, the features that represent **95%** **of the total importance score** will be retained.

The next figure displays the importance scores for each feature (in descending order of importance). For instance, the most important feature received a score of 27.1% while the least important feature received an almost irrelevant score of 0.02%.

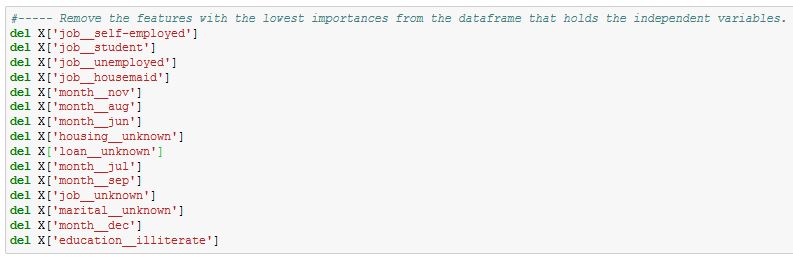
The last column displays the cumulative importance scores. It can be used to determine how many features should be kept according to their importance in predicting the outcomes. The values indicate that **35 features** will need to be retained in order to preserve 95% of the total importance score. This will reduce the count of features from 50 to 35 without losing valuable feature-related information required to interpret the results.



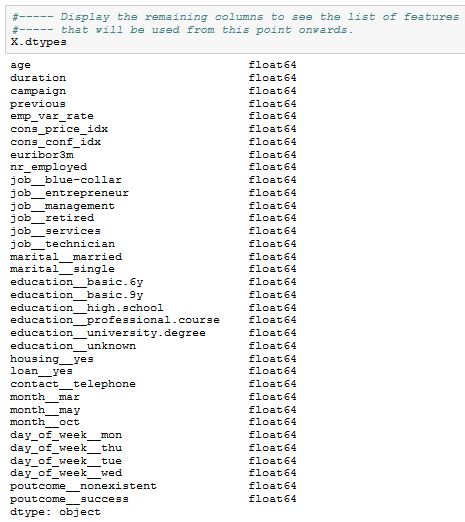
# 5. Remove the Least Important Features

The *Extra Trees Classifier* decision tree has shown that we can remove 15 features that represent a combined total of less than 5% of the importance scores.

Remove the 15 features from the dataset:



Display the remaining list of features that will be used to build the predictive models:

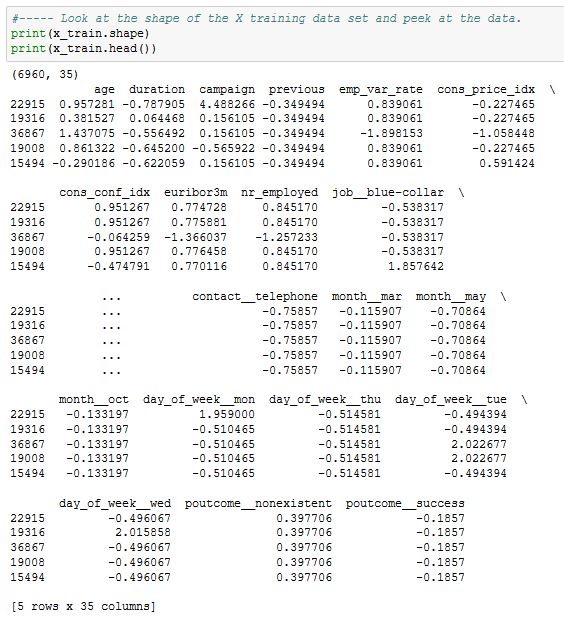


# 6. Split the Dataset into Training and Testing Sets

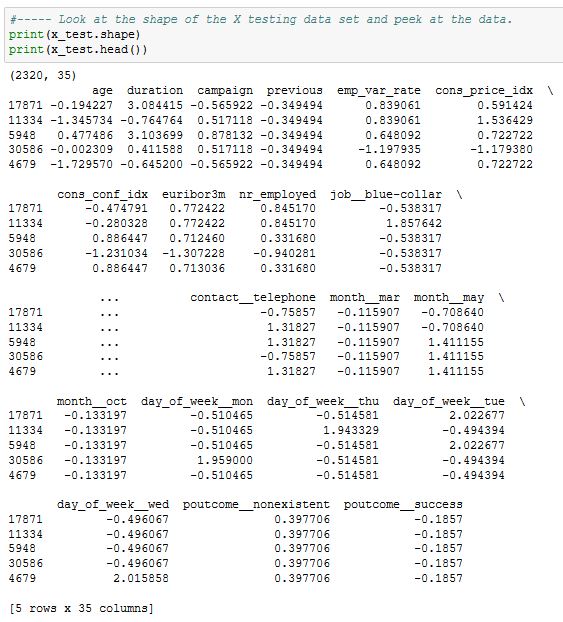
Split the data into training and testing sets using a 75/25 ratio:



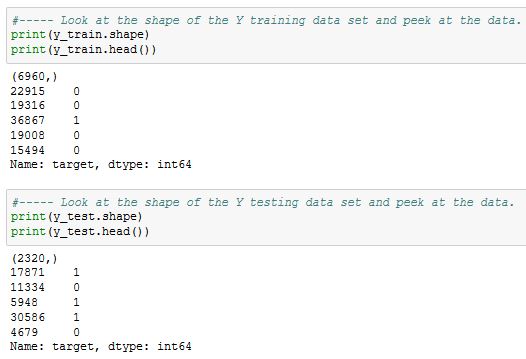
Look at the shape and contents of the X training data to confirm that the process was successful. The following snapshot shows that the training data is comprised of 6960 observations with 35 features:



Look at the shape and contents of the X testing data to confirm that the process was successful. The following snapshot shows that the training data is comprised of 2320 observations with 35 features:



Look at the shape and contents of the Y training and testing data to confirm that the process was successful. The following snapshot confirms that the Y data has the same number of rows as for the X data:



# 7. Summary

The *Primary Component Analysis* technique was applied to the dataset but it was found to offer minimal benefits and would have resulted in the loss of valuable feature-related information required to interpret the results. As an alternative, an *Extra Trees Classifier* decision tree was used. The count of features was reduced from 50 to 35 with a minimal loss of information.

Finally, the dataset was split into training and testing sets.

The dataset is now ready to be used by the model building process.

# Appendix A – Listing of Python Commands Used for this Report

#----- Import the libraries and functions that will be used

import pandas as pd

from pandas import Series, DataFrame

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

from sklearn.preprocessing import StandardScaler

from sklearn.decomposition import PCA

from sklearn.model\_selection import train\_test\_split

from sklearn.metrics import classification\_report, accuracy\_score, confusion\_matrix

from sklearn.ensemble import ExtraTreesClassifier

%matplotlib inline

**Separate the Dataset Features into X and Y Datasets**

#----- Create a dataframe called 'X' to hold only the model’s independent variables.

X = balanced\_df.copy()

del X['target'] # Remove the target variable.

#----- Create a dataframe called 'Y' to hold only the model’s target variable.

Y = balanced\_df['target']

**Perform Primary Component Analysis on the Dependent Variables**

#----- Perform the Principal Component Analysis on the balanced dataset of dependent variables.

pca = PCA(n\_components = 50)

sklearn\_pca\_X = pca.fit\_transform(X)

#----- The goal is find out how many how many components would be enough to explain almost all of the

#----- data variance. First, the covariance matrix must be generated.

covariance\_matrix = np.cov(X.T)

#----- Display the covariance matrix to confirm that it was generated successfully.

print(covariance\_matrix)

#----- Use the covariance matrix to calculate the Eigen Vectors and Eigen Values.

eig\_vals, eig\_vecs = np.linalg.eig(covariance\_matrix)

#----- Display the Eigen Values to confirm that they were generated successfully.

print(eig\_vals)

#----- Display the Eigen values (in descending order) as well as their percentages of variance and cumulative percentages.

print("\n Eigen value\t\tPercentage of Variance\t\tCumulative Percentage\n")

counter = 1

cumulative\_pct = 0.0

for i in list(np.sort(eig\_vals)[::-1]):

cumulative\_pct = cumulative\_pct + (i/sum(eig\_vals)\*100)

print("%d\t%.3f\t\t\t%.3f\t\t\t\t%.3f" %

(counter,i,(i/sum(eig\_vals)\*100),cumulative\_pct))

counter = counter + 1

#----- Plot the variance ratios for each component. This will show us where the variance contributions become minimal.

explained\_variance = pca.explained\_variance\_ratio\_

plt.plot(explained\_variance,color='Blue', linewidth=3)

plt.rcParams['figure.figsize'] = [9, 9]

plt.title('Variance Ratios\n',size=30)

plt.ylabel('Variance Contributions\n',size=25)

plt.xlabel('\nNumber of Components',size=25)

plt.xticks(fontsize=14)

plt.yticks(fontsize=14)

**Estimate the Importance of Features using Extra Trees Classifier Decision Tree**

#----- Build a forest and compute the importance of each feature.

forest = ExtraTreesClassifier(n\_estimators=250, random\_state=0)

forest.fit(X,Y)

#----- Store the importance values and display them to confirm that they have been generated.

importances = forest.feature\_importances\_

print(importances)

#----- Sort the array elements by the importance values, in descending order.

indices = np.argsort(importances)[::-1]

#----- Display the features and their importance values, in descending order of importance.

print("\n Importance Cumulative")

print(" Importance")

print("----------------------------------------------------------------------------------------")

cumulative\_pct = 0

for i in range(0,X.shape[1]):

cumulative\_pct += importances[indices[i]]

print("%d\tFeature %d: %30s\t\t%0.2f\t\t%0.2f" % (i+1, indices[i], X.columns.values[indices[i]],

importances[indices[i]]\*100, cumulative\_pct\*100))

#----- Plot the feature importances, in descending order of importance.

plt.figure(figsize=(15,10))

plt.title("\nImportance Levels for Each Feature\n", size=25)

plt.ylabel('Importance Level\n',size=25)

plt.xlabel('\nFeature Number',size=25)

plt.bar(range(X.shape[1]), importances[indices], color="green", align="center")

plt.xticks(range(X.shape[1]), indices, size=10)

plt.xlim([-1, X.shape[1]])

plt.show()

**Remove the Least Important Features**

#----- Remove the features with the lowest importances from the dataframe that holds the independent variables.

del X['job\_\_self-employed']

del X['job\_\_student']

del X['job\_\_unemployed']

del X['job\_\_housemaid']

del X['month\_\_nov']

del X['month\_\_aug']

del X['month\_\_jun']

del X['housing\_\_unknown']

del X['loan\_\_unknown']

del X['month\_\_jul']

del X['month\_\_sep']

del X['job\_\_unknown']

del X['marital\_\_unknown']

del X['month\_\_dec']

del X['education\_\_illiterate']

#----- Display the remaining columns to see the list of features

#----- that will be used from this point onwards.

X.dtypes

**Split the Dataset into Training and Testing Sets**

#----- Split the data into training/testing sets using a 75/25 ratio.

x\_train, x\_test, y\_train, y\_test = train\_test\_split(X, Y, test\_size=0.25, random\_state=1)

#----- Look at the shape of the X training data set and peek at the data.

print(x\_train.shape)

print(x\_train.head())

#----- Look at the shape of the X testing data set and peek at the data.

print(x\_test.shape)

print(x\_test.head())

#----- Look at the shape of the Y training data set and peek at the data.

print(y\_train.shape)

print(y\_train.head())

#----- Look at the shape of the Y testing data set and peek at the data.

print(y\_test.shape)

print(y\_test.head())