AIRLINE PROJECT

AIT-582: Applications of Metadata in Complex Big Data Problems

FINAL PROJECT REPORT

Under the guidance of:

Dr. Setareh Rafatirad

Submitted By:

Pranav Makhijani

prnvmakhijani45@gmail.com

Introduction & Problem Formulation

One of the most important quality of a good Data Scientist is to have excellent Feature Engineering skills. Statistics have showed that a they spend almost 70% of the time in getting the data in the right format to feed to Machine Learning models. There are a number of steps to be performed in order to get data ready for the modeling phase. Once these steps are completed, only then can we move to Data Modeling and Visualizations. The order of the steps are –

1. Data Extraction
2. Data Cleaning
3. Exploratory Data Analysis
4. Feature Engineering
5. Dimensionality Reduction
6. Transformation & Scaling

Following the above steps, the Airline Project focuses heavily on six steps mentioned above. The customer database which has to be analyzed to understand why some customers are flying your airlines while others are canceling.

This project has been done completely in Python because of the wide variety of libraries it has to offer. The IDE used is Spyder which is available with the Anaconda distribution. The following libraries have been used for this project -

Data Extraction	json, csv, urlopen
Data Manipulation	pandas, numpy
Data Visualization	matplotlib, seaborn
Data Modeling	scikit-learn
Scikit-Learn libraries	label encoder, one hot encoder, standard scaler, train test split, PCA, confusion matrix, classification report, SVC, cross validation score, K-neighbors classifier, grid search
Deep Learning	Keras, tensorflow

Note: I had to use Python version 3.5 to run Deep Learning algorithm Tensor Flow since it is available only with that version on Windows. Keras is a wrapper above the Tensor Flow library.

Dataset Description

The data is provided on a URL - http://ist.gmu.edu/~hpurohit/courses/ait582-proj-data-spring16.json from which it has to be programmatically downloaded. The data is in the JSON format and consists of 10891 rows and 6 columns. Here is the glimpse of the dataset after converting it into a data frame:

Index	GUESTS	SUCCESS	SEATCLASS	DESCRIPTION	CUSTOMERID	FARE
0	1	0	3	Braund, Mr. Owen Harris;22	1	7.25
1	1	1	1	Cumings, Mrs. John Bradley …	2	71.3
2	0	1	3	Heikkinen, Miss. Laina;26	3	7.92
3	1	1	1	Futrelle, Mrs. Jacques Heath…	4	53.1
4	0	0	3	Allen, Mr. William Henry…	5	8.05
5	0	0	3	Moran, Mr. James;	6	8.46
6	0	0	1	McCarthy, Mr. Timothy J;54	7	51.9
7	3	0	3	Palsson, Master. Gosta…	8	21.1
8	0	1	3	Johnson, Mrs. Oscar W (Elis…	9	11.1
9	1	1	2	Nasser, Mrs. Nicholas (Ade…	10	30.1
10	1	1	3	Sandstrom, Miss. Marguer…	11	16.7

Here is a brief description of the features:

1. CUSTOMERID: A unique ID associated to a customer.
2. GUESTS: Number of guests accompanying the customer.
3. SUCCESS: Categorical variable which displays whether customer travelled or not.
4. SEATCLASS: Categorical variable which displays the seat class of the customer.
5. DESCRIPTION: Description of the customer including his name and age
6. FARE: Total fare paid by the customer

The features of this dataset are Guests, Seat Class, Description and Fare. The target variable is Success.

The train-test ratio for this project is 0.8:0.2. The reason for such a high training set ratio is usage of Neural Networks which tend to perform better with higher volumes of data.

Project Milestone 1: Metadata Extraction & Imputation:

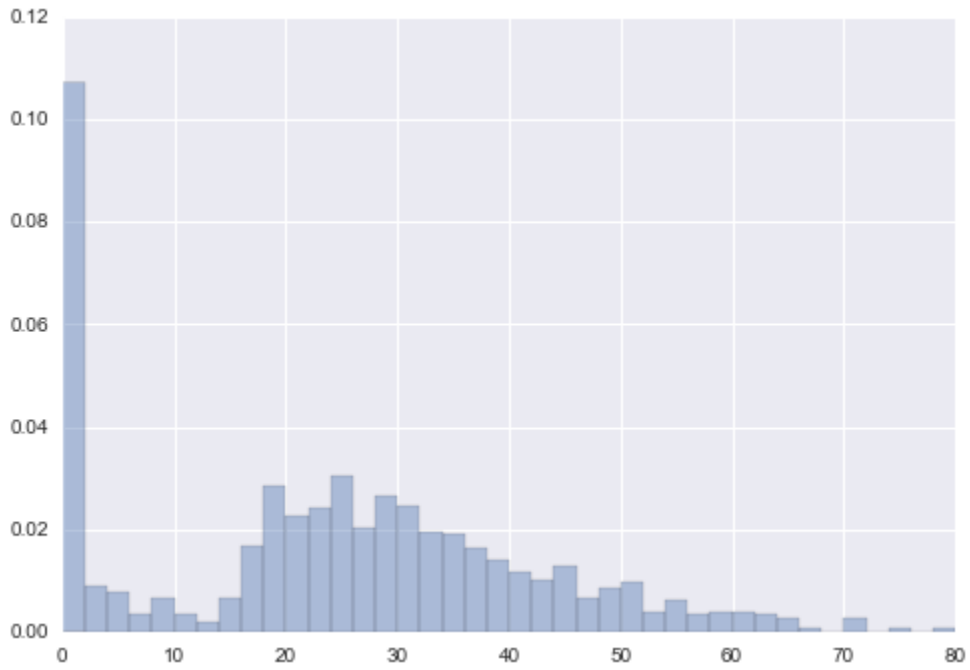
The next step after extraction of data is to observe the dataset and look for opportunities to derive semantic metadata. This is where Feature Engineering skills are very useful. The DESCRIPTION feature of the customer consists of his age and name prefix. Subsequently, it is easy to derive the gender of the customer from the name prefix. It is very interesting to see the role of these new features during the Data Modeling phase.

Here is how the dataset looks after Metadata Extraction:

Index	DESCRIPTION	SUCCESS	GUESTS	SEATCLASS	CUSTOMERID	FARE	AGE	TITLE	GENDER
0	Braund, Mr. Owen Harris;22	0	1	3	1	7.25	22	Mr	Male
1	Cumings, Mrs. John Bradley …	1	1	1	2	71.3	38	Mrs	Female
2	Heikkinen, Miss. Laina;26	1	0	3	3	7.92	26	Miss	Female
3	Futrelle, Mrs. Jacques Heath…	1	1	1	4	53.1	35	Mrs	Female
4	Allen, Mr. William Henry…	0	0	3	5	8.05	35	Mr	Male
5	Moran, Mr. James;	0	0	3	6	8.46	0	Mr	Male
6	McCarthy, Mr. Timothy J;54	0	0	1	7	51.9	54	Mr	Male

The result of the above step is addition of three new features to our dataset – AGE, TITLE & GENDER.

The Age of some customers is missing in the dataset. A good example of this would be Customer ID - 6 in the above case. It makes sense to observe the number of customers with missing age. The capabilities of seaborn library can be used for this purpose.



It can be safely interpreted from the above visualization that Age for many customers is missing. It is ideal to come up with a suitable data imputation method for this scenario.

There are several data imputation methods like mean imputation, median imputation, KNN imputation, etc. There is a need to use existing knowledge about the data, make assumptions and validate them using querying and visualization.

Assumption 1: Median age varies with prefix title of the customer.

The median age of the customers is 24. The prefix title of customers with missing age are Mr, Mrs, Miss, Master & Dr. Median imputation cannot be used in this case because imputing median age of 24 for a customer with title Master is incorrect since it is assumed that a person with that title is under 18. Similarly, it is safe to assume that a customer with title Dr is above 24 is majority circumstances. The best way to validate the assumptions would be to query the data.

```
In [11]: airline[airline["TITLE"] == " Mr"]["AGE"].median()
Out[11]: 25.0

In [12]: airline[airline["TITLE"] == " Mrs"]["AGE"].median()
Out[12]: 33.0

In [13]: airline[airline["TITLE"] == " Miss"]["AGE"].median()
Out[13]: 18.0

In [14]: airline[airline["TITLE"] == " Master"]["AGE"].median()
Out[14]: 3.0

In [15]: airline[airline["TITLE"] == " Dr"]["AGE"].median()
    ...:
Out[15]: 44.0
```

Following things can be interpreted from the above query results:

1. Median age of a customer with title Mr is 25.
2. Median age of a customer with title Mrs is 33.
3. Median age of a customer with title Miss is 18.
4. Median age of a customer with title Master is 3.
5. Median age of a customer with title Dr is 44.

Hence, Assumption 1 has been successfully validated.

Assumption 2: Age of customers travelling in 1st class is higher than that of other classes.

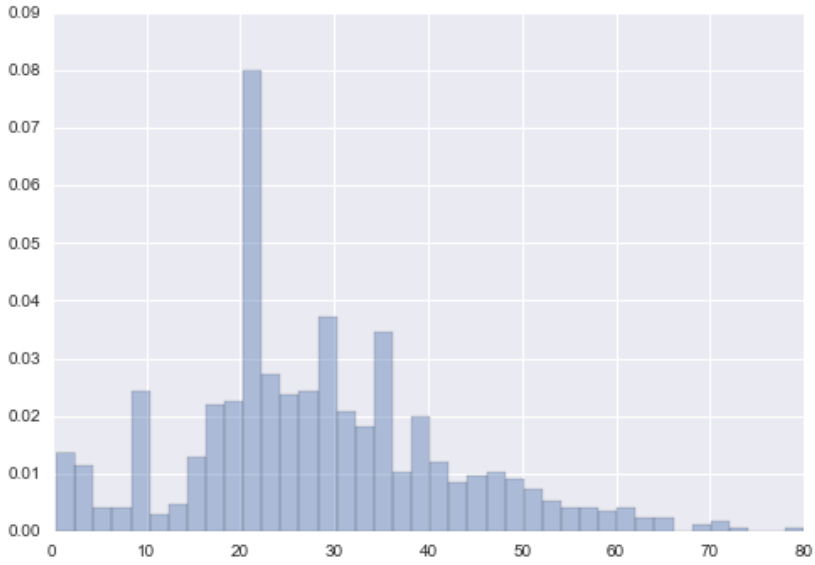
The next step would be to validate our assumption.

```
In [17]: airline[airline["SEATCLASS"] == 1]["AGE"].median()
Out[17]: 35.0

In [18]: airline[airline["SEATCLASS"] == 2]["AGE"].median()
Out[18]: 28.0

In [19]: airline[airline["SEATCLASS"] == 3]["AGE"].median()
Out[19]: 20.0
```

Here is the output of the above query results:

1. Median age of customers travelling in 1st class in 35.
2. Median age of customers travelling in 2nd class in 28.
3. Median age of customers travelling in 3rd class in 20.

There is a steady decline in age as the passenger class goes down. Since both the assumptions have been successfully validated by the dataset, their combination can be used to come up with a suitable data imputation method. Hence, the combination of passenger class and prefix title is used for imputing missing age. Here is the distribution of the AGE feature after applying the imputations. The distribution is close to normal now.



The FARE of some customers is missing. It is suitable to come up with assumptions and validate them by querying data for imputing missing values.

Assumption 1: Passengers travelling in higher class have higher fare.

This is a very straightforward assumption can be easily validated by the data.

```
In [24]: airline[airline["SEATCLASS"] == 1]["FARE"].mean()
Out[24]: 84.15468749999992

In [25]: airline[airline["SEATCLASS"] == 2]["FARE"].mean()
Out[25]: 20.66218315217391

In [26]: airline[airline["SEATCLASS"] == 3]["FARE"].mean()
Out[26]: 13.675550101832997
```

The above query results clearly state:

1. Mean fare of customers travelling in 1st class in 84.15
2. Mean fare of customers travelling in 2nd class in 20.66
3. Mean fare of customers travelling in 3rd class in 13.67

Assumption 2: Customers travelling with more guests pay higher fare.

Here is how the assumption can be validated

1. Mean fare of customer travelling with no guests is 25.69
2. Mean fare of customer travelling with 1 guest is 44.14
3. Mean fare of customer travelling with 2 guests is 51.75
4. Mean fare of customer travelling with 3 guests is 68.90

The combination of both the assumptions – passenger class and guests can be used to impute missing fare values.

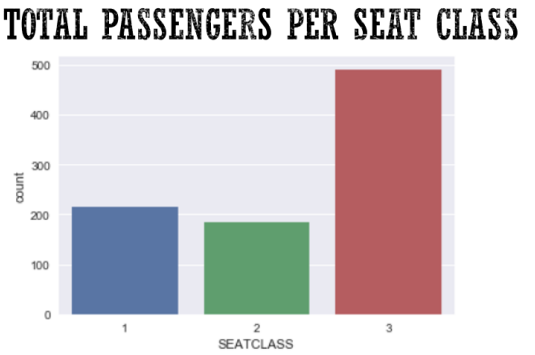
Project Milestone 2: Summarization & Visualization:

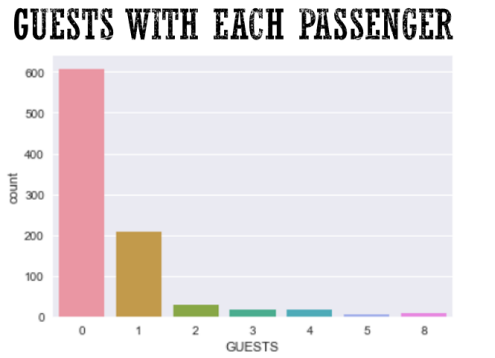
The next step after imputing missing values is Exploratory Data Analysis. This is a very important step because a lot of information about data can be obtained through Visualizations. All visualizations in this project have been done using matplotlib and seaborn libraries of Python.

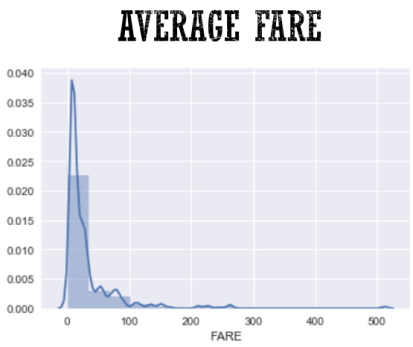
Here is a statistical overview of all numerical features:

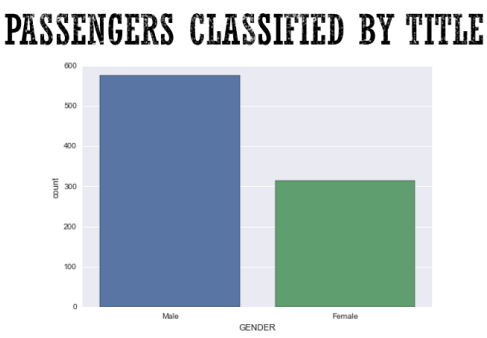
	GUESTS	SEATCLASS	FARE	AGE
count	891.000000	891.000000	891.000000	891.000000
mean	0.523008	2.308642	32.783674	28.256083
std	1.102743	0.836071	49.664932	13.919674
min	0.000000	1.000000	4.012500	0.420000
25%	0.000000	2.000000	7.925000	21.000000
50%	0.000000	3.000000	14.500000	26.000000
75%	1.000000	3.000000	31.275000	36.000000
max	8.000000	3.000000	512.329200	80.000000

Visualizations will give a much better description of data than the above statistics.









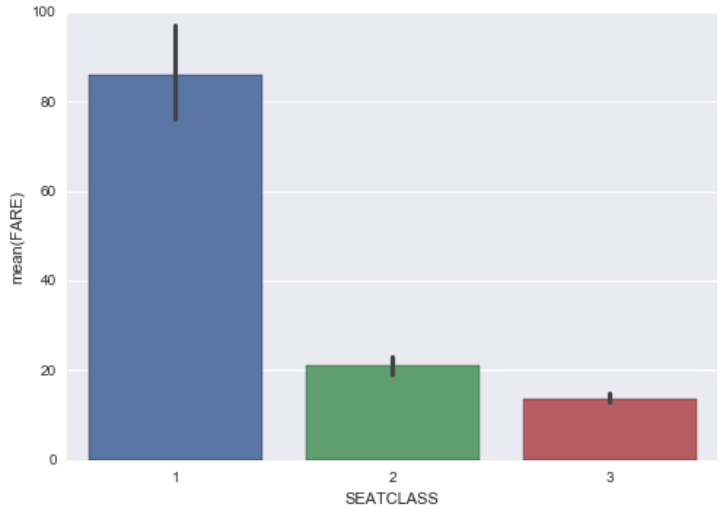
Here are the results of the above visualizations:

1. Majority of the people are travelling in the 3rd class. The surprising part is customers travelling in 1st class are more than those travelling in 2nd class.
2. A significant part of the customers are travelling alone.
3. The average fare paid by a customer is less than $40. This makes sense because of the first point stating that majority of the people travel by 3rd class.

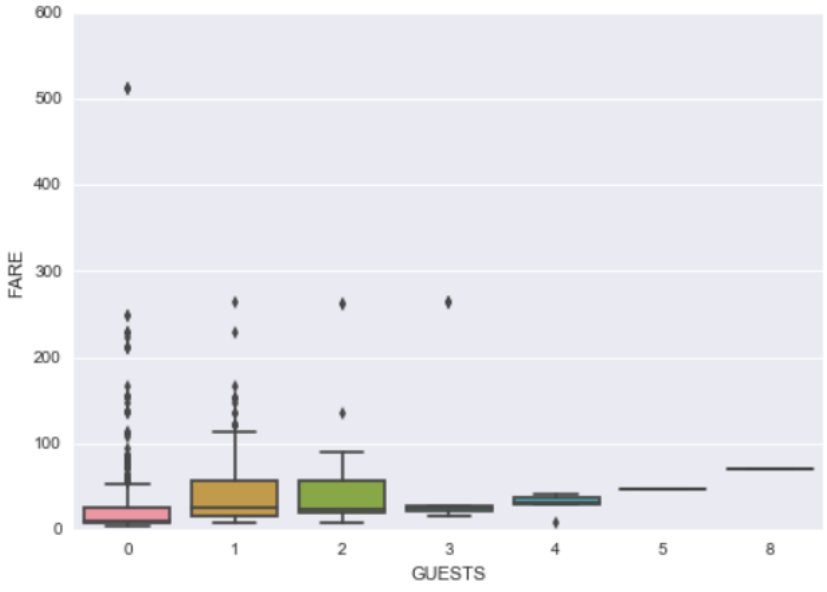
4. It’s a male dominated airline.

Some other visualizations:

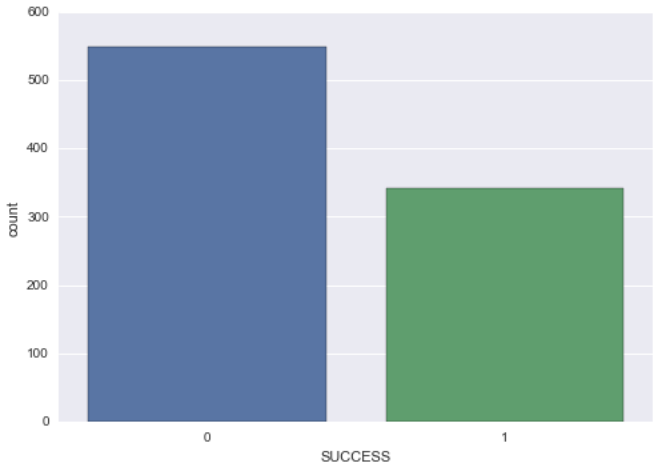
1. Mean fare of passengers travelling in each class:



2. Box-Plot of fares paid by customers travelling with 0-8 guests:



3. Higher number of customers cancel the flights



Principal Component Analysis

PCA is a very important step in the Machine Learning cycle. It is a very strong tool which represents all the features in user-defined-number-of-components. It is possible to define all the features on the Airline dataset in two principal components and plot them on a 2D graph to get a better understanding of the problem. The components are chosen in such a way that they explain the maximum variance of the dataset.

The goal of PCA is to try to depict and visualize all features in the form of two principal components.

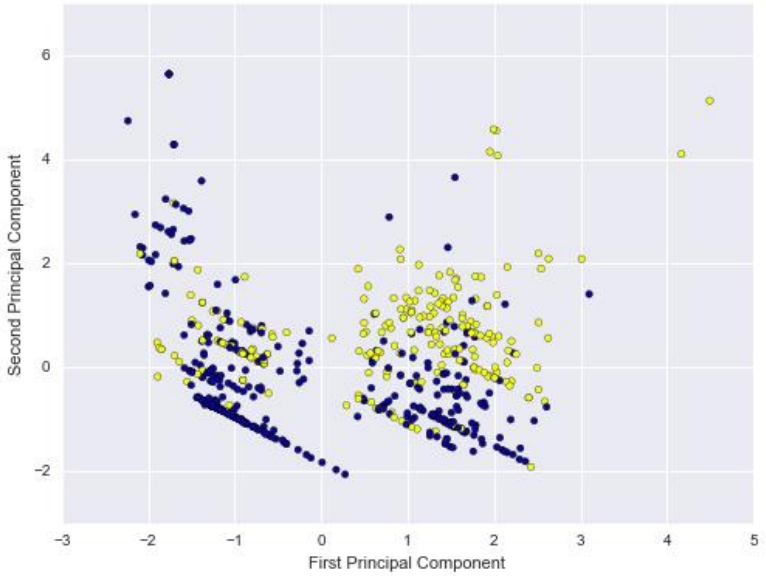
	0	1	2	3	4	5
PC-1	0.452531	-0.683457	-0.155894	-0.126480	0.347256	0.408920
PC-2	-0.116612	-0.047280	-0.489664	0.623192	0.481326	-0.352636

Here is the output of variance of all features in the form of two components. It can be interpreted as:

Principal Component 1 = (0.452531 * Feature 0) + (-0.683457 * Feature 1) + … + (0.408920 * Feature 5)

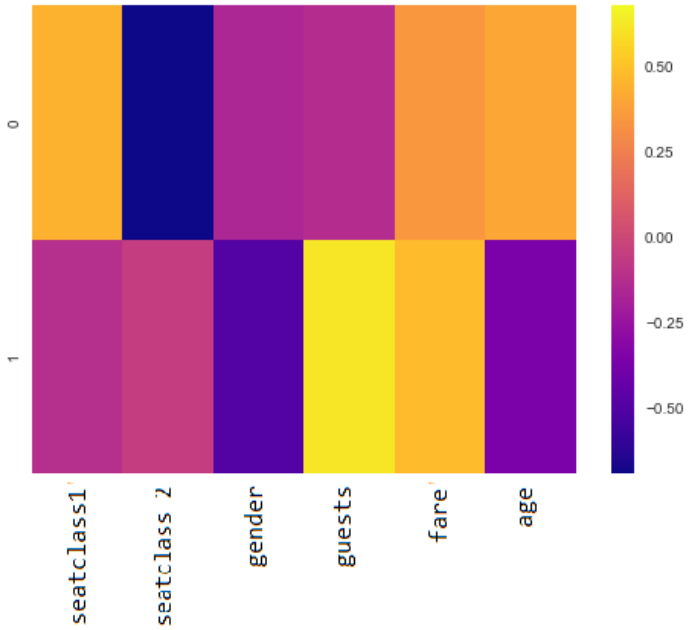
Principal Component 2 = (-0.116612 * Feature 0) + (-0.047280 * Feature 1) + … + (-0.352636 * Feature 5)

Here is a plot of Principal Component 1 vs Principal Component 2:



A colormap has been used to distinguish the principal components by the dependent variable – SUCCESS. The clusters can be quite significantly distinguished on some parts of the graph and overlap on the other parts. But this graph does a very good job at representing all features on a 2D scale and distinguishing them by the dependent variable.

The below figure depicts all the correlation of all six features to the principal components. The principal components are depicted as rows and features as columns. The brighter the color in terms of principal components, the higher the variance is explained by the variables.



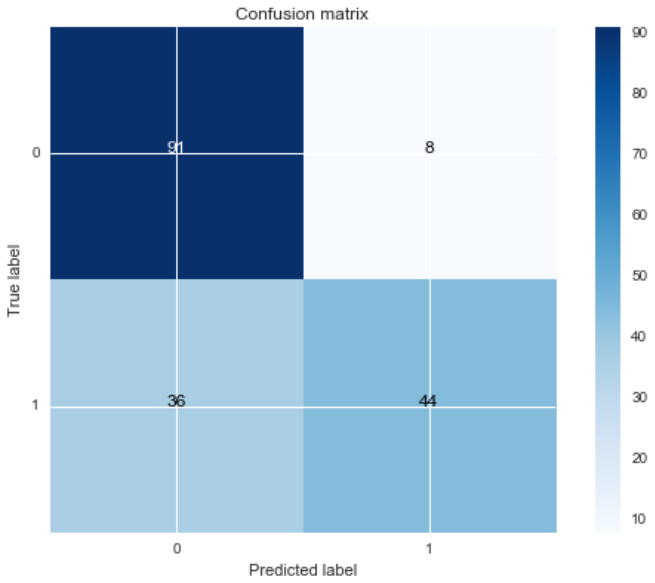
Project Milestone 3 & 4: Analytics & Data Modeling

The data is now prepared in the right format to feed to Machine Learning algorithms. The model can be run on the entire dataset instead of selecting important variables since there are only six features in this dataset. The aim of this phase is to run multiple algorithms and chose the best performing one.

Random Forest Classifier:

A Random Forest is an ensemble of multiple decision trees in which features and data are chosen at random. After running the Random Forest classifier with 10 trees, here is the output:

Confusion Matrix:



Classification Report:

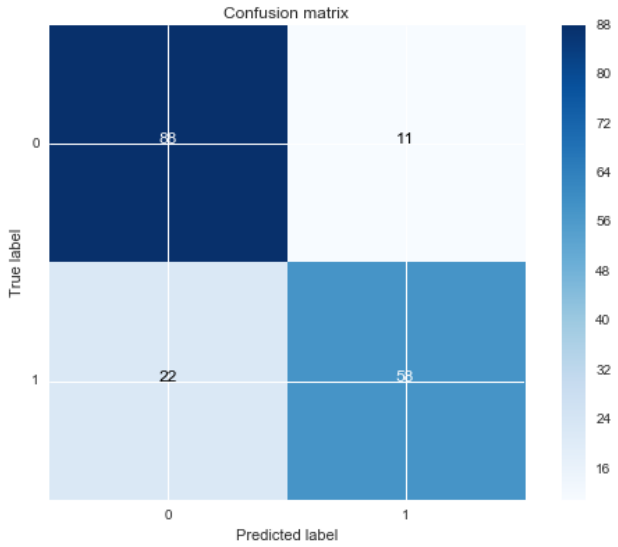
	precision	recall	f1-score	support
0	0.72	0.92	0.81	99
1	0.85	0.55	0.67	80
avg / total	0.77	0.75	0.74	179

Accuracy achieved on test set is 75.41%

KNN Classifier:

KNN classification works on the following principle – it classifies a new point based on the majority vote based on k nearest neighbors. Taking the value of k=5, here is the output of KNN classifier:

Confusion Matrix:



Classification Report:

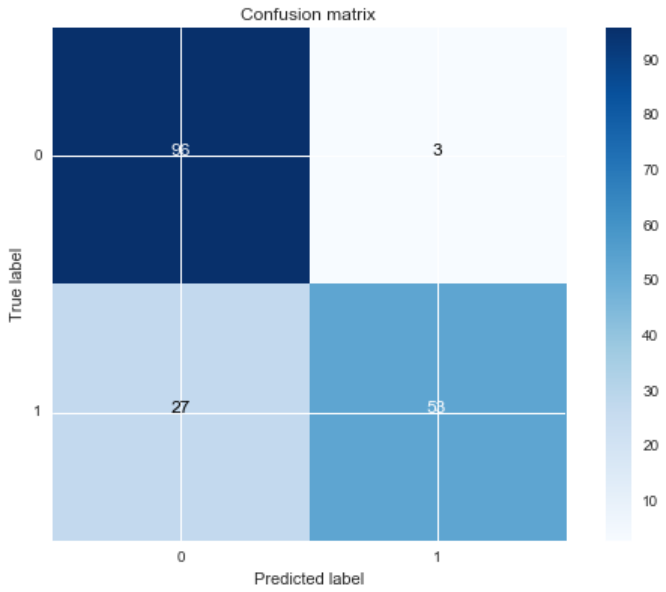
	precision	recall	f1-score	support
0	0.80	0.89	0.84	99
1	0.84	0.72	0.78	80
avg / total	0.82	0.82	0.81	179

Accuracy on the test set: 81.56%

Deep Neural Network:

A deep neural network (DNN) is an artificial neural network (ANN) with multiple hidden layers of units between the input and output layers. Similar to shallow ANNs, DNNs can model complex non-linear relationships. In this case, the deep neural network is implemented using Keras libraries with Tensor Flow backend. The neural network has two hidden layers and that is why it is referred to as a Deep Neural Network. Here is its output –

Confusion Matrix:



Classification Report:

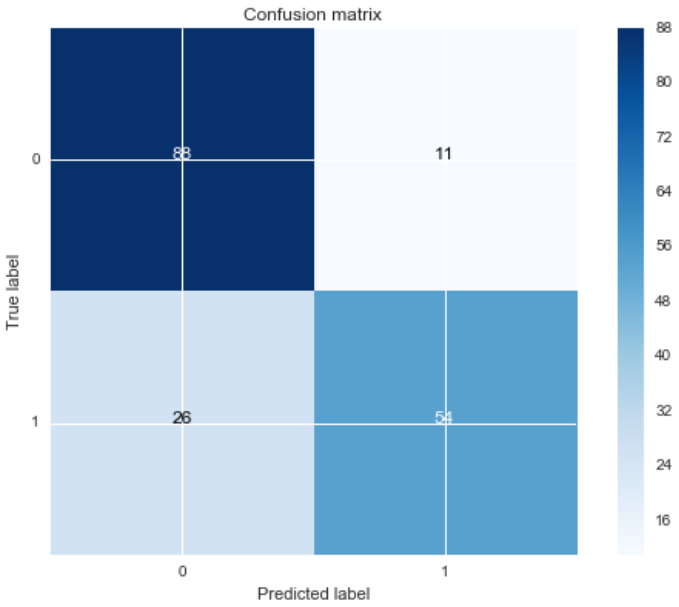
	precision	recall	f1-score	support
0	0.78	0.97	0.86	99
1	0.95	0.66	0.78	80
avg / total	0.85	0.83	0.83	179

Accuracy on test set: 83.24%

Support Vector Machine Classifier:

Support Vector Machines are used both for Regression & Classification. The SVM used for Classification is called Support Vector Classifier. The aim is to run the SVM classifier with general parameters and improve the accuracy using hyperparameter tuning. Running the classifier with minimum parameters and linear kernel, the output is:

Confusion Matrix:

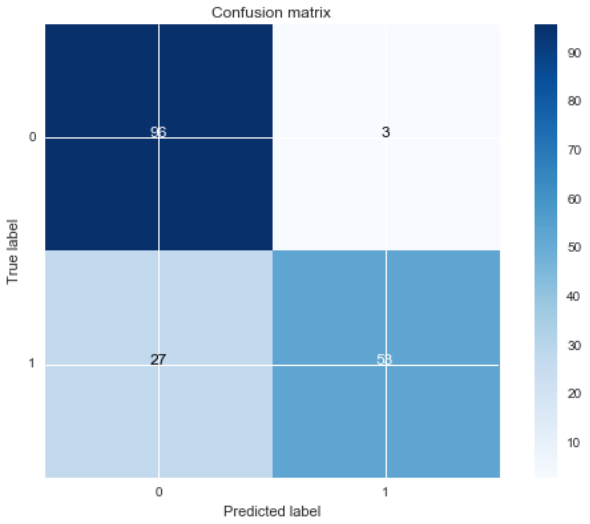


Accuracy on training set is 79.32%

The aim now is to increase accuracy using hyperparameter tuning. One efficient technique to do the same is Grid Search. After running Grid Search, following parameters are identified to be the best parameters:

Key	Type	Size	Value
C	int	1	10
gamma	float	1	0.1
kernel	str	1	rbf

Re-running the SVM Classifier model with the above parameters will give us a better result. Here is the output –

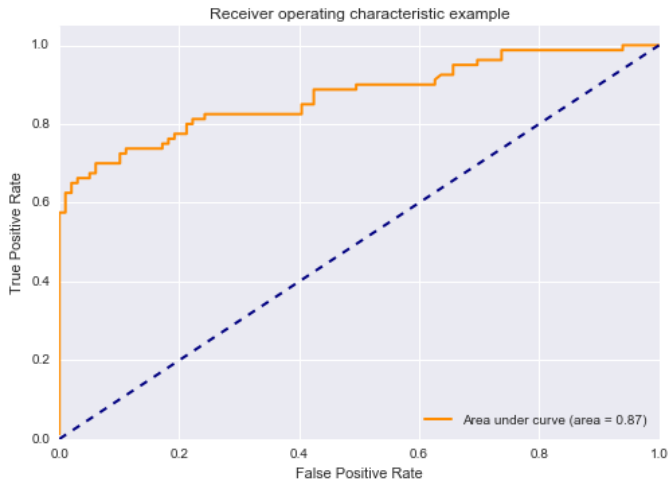


Accuracy on the test set is 83.24% which is an improvement of close to 4%.

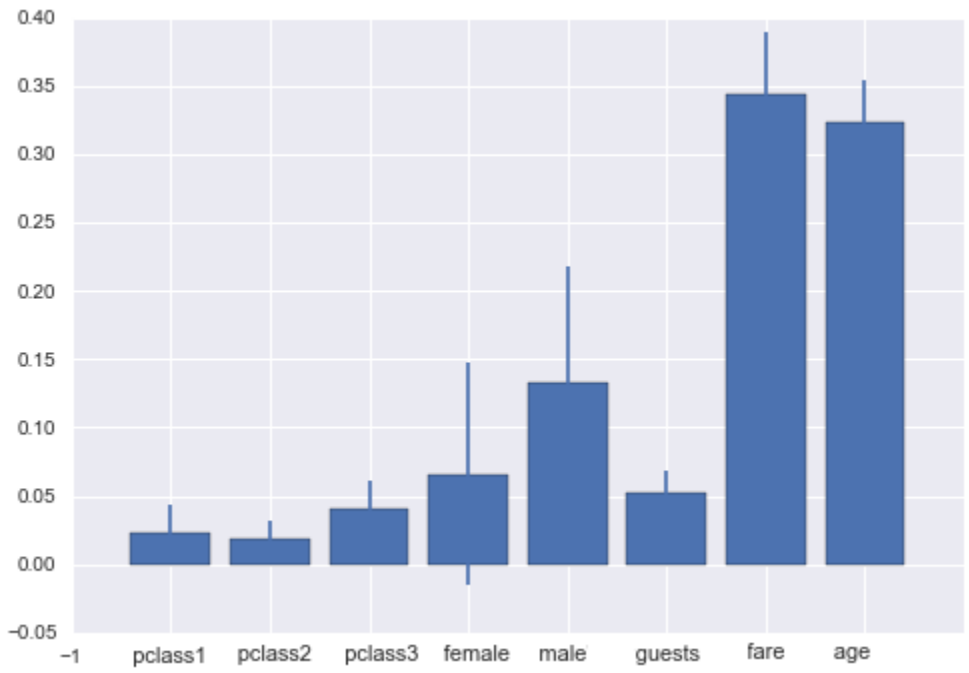
Classification Report:

	precision	recall	f1-score	support
0	0.78	0.97	0.86	99
1	0.95	0.66	0.78	80
avg / total	0.85	0.83	0.83	179

Plotting the ROC curve for SVM:



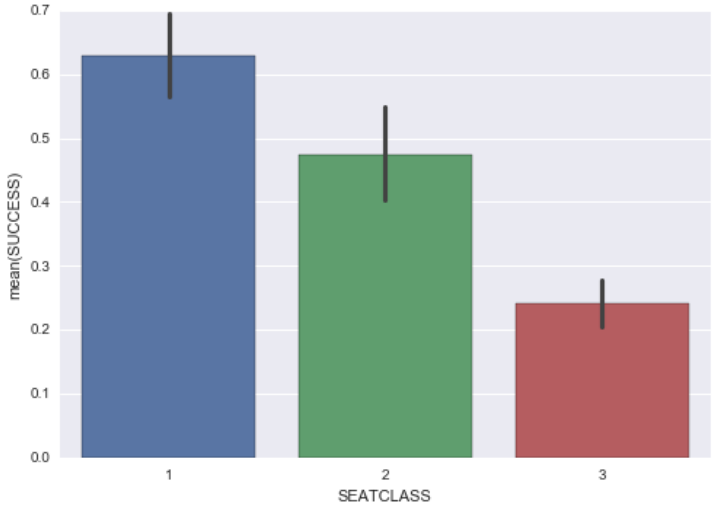
Variable Importance Plot:



Data Insights:

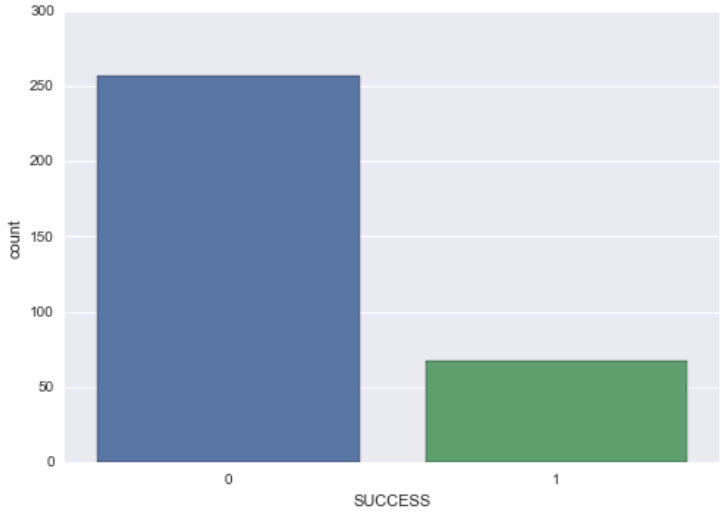
Here are some of the data insights that can be made after modeling and visualizing the Airline dataset. A deep dive into these facts should be done by the social media team to devise profitable strategies in the future.:

1. Success Ratio in Passenger Class 3 is less than 25%



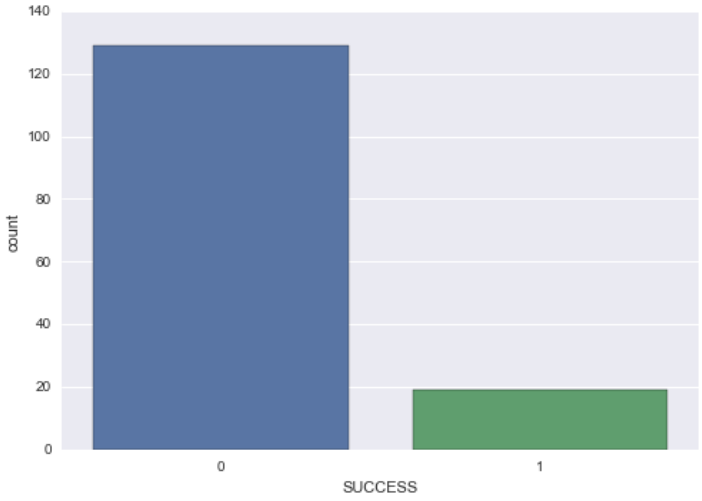
2. 4 out of 5 passengers cancel their tickets if their ticket fare is less than $10.

```
In [303]: sns.countplot(pclass[pclass["FARE"] < 10]["SUCCESS"])
Out[303]: <matplotlib.axes._subplots.AxesSubplot at 0x21e1e08ee80>
```



3. 5 out of 6 passengers aged between 20 and 25 cancel their tickets.

```
In [306]: sns.countplot(pclass[(pclass["AGE"] > 20) & (pclass["AGE"] < 25)]["SUCCESS"])
Out[306]: <matplotlib.axes._subplots.AxesSubplot at 0x21e1f0fbb70>
```



RDF Output:

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE rdf:RDF [     <!ENTITY rdf 'http://www.w3.org/1999/02/22-rdf-syntax-ns#'>
        <!ENTITY rdfs 'http://www.w3.org/2000/01/rdf-schema#'>
        <!ENTITY xsd 'http://www.w3.org/2001/XMLSchema#'>
]>
<rdf:RDF xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#" xmlns:xsd="http://www.w3.org/2001/XMLSchema#" xmlns="http://example.org/data/AirlinePreprocess.csv#"
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#">
  <rdf:Description rdf:about="http://example.org/data/AirlinePreprocess.csv#row=1">
    <AGE rdf:datatype="&xsd;integer">22</AGE>
    <FARE rdf:datatype="&xsd;decimal">7.25</FARE>
    <GENDER>Male</GENDER>
    <GUESTS rdf:datatype="&xsd;integer">1</GUESTS>
    <SEATCLASS rdf:datatype="&xsd;integer">3</SEATCLASS>
    <SUCCESS>0</SUCCESS>
    <TITLE xml:lang="en-ca"> Mr</TITLE>
  </rdf:Description>
  <rdf:Description rdf:about="http://example.org/data/AirlinePreprocess.csv#row=10">
    <AGE rdf:datatype="&xsd;integer">14</AGE>
    <FARE rdf:datatype="&xsd;decimal">30.0708</FARE>
    <GENDER>Female</GENDER>
    <GUESTS rdf:datatype="&xsd;integer">1</GUESTS>
    <SEATCLASS rdf:datatype="&xsd;integer">2</SEATCLASS>
    <SUCCESS>1</SUCCESS>
    <TITLE xml:lang="en-ca"> Mrs</TITLE>
  </rdf:Description>
  <rdf:Description rdf:about="http://example.org/data/AirlinePreprocess.csv#row=100">
    <AGE rdf:datatype="&xsd;integer">34</AGE>
    <FARE rdf:datatype="&xsd;decimal">26</FARE>
    <GENDER>Male</GENDER>
    <GUESTS rdf:datatype="&xsd;integer">1</GUESTS>
    <SEATCLASS rdf:datatype="&xsd;integer">2</SEATCLASS>
    <SUCCESS>0</SUCCESS>
    <TITLE xml:lang="en-ca"> Mr</TITLE>
  </rdf:Description>
  <rdf:Description rdf:about="http://example.org/data/AirlinePreprocess.csv#row=101">
    <AGE rdf:datatype="&xsd;integer">28</AGE>
    <FARE rdf:datatype="&xsd;decimal">7.8958</FARE>
    <GENDER>Female</GENDER>
    <GUESTS rdf:datatype="&xsd;integer">0</GUESTS>
    <SEATCLASS rdf:datatype="&xsd;integer">3</SEATCLASS>
    <SUCCESS>0</SUCCESS>
    <TITLE xml:lang="en-ca"> Miss</TITLE>
  </rdf:Description>
```

Tools used:


python
Pandas
matplotlib
scikit
learn
Seaborn
spyder
NumPy
TensorFlow

References:

1. Ayoub, Shahid. How To Parse and Convert JSON to CSV using Python. October 15, 2015. Applied Informatics. http://blog.appliedinformaticsinc.com/how-to-parse-and-convert-json-to-csv-using-python/
2. Albon, Chris. Using Seaborn To Visualize A Pandas Dataframe. May 01, 2016. https://chrisalbon.com/python/pandas_with_seaborn.html
3. Elite Data Science. Keras Tutorial: The Ultimate Beginner's Guide to Deep Learning in Python. November 25, 2016. https://elitedatascience.com/keras-tutorial-deep-learning-in-python
4. Ray, Sunil. Understanding Support Vector Machine algorithm from examples (along with code). Analytics Vidhya. October 6, 2015. https://www.analyticsvidhya.com/blog/2015/10/understaing-support-vector-machine-example-code/
5. Brownlee, Jason. How to Grid Search Hyperparameters for Deep Learning Models in Python With Keras. Machine Learning Mastery. http://machinelearningmastery.com/grid-search-hyperparameters-deep-learning-models-python-keras/
6. Raschka, Sebastian. Implementing a Principal Component Analysis (PCA) – in Python, step by step. April 13, 2014. http://sebastianraschka.com/Articles/2014_pca_step_by_step.html