Bank Marketing

Chinmay Sathe

19MT0119

—

NEURAL NETWORKS & DEEP LEARNING LAB
(MCC542)

—

Prof. Subhashis Chatterjee

Abstract

Most banks prefer calling each and every customer for advertising their products. Thus, using brute force technique to sell their product. For the same they have to spent a lot of money and time. But what if they already knew if a customer will take the product (term deposit) or not.

Objective here is to develop a Machine Learning algorithm that predicts if the client will subscribe a term deposit or not.

1 INTRODUCTION

A term deposit is a type of deposit account held at a financial institution where money is locked up for some set period of time. Term deposits are usually short-term deposits with maturities ranging from one month to a few years.

Typically, term deposits offer higher interest rates than traditional liquid savings accounts whereby customers can withdraw their money at any time.

Here, we are thinking from bank's perspective. We want to increase bank's profit by encouraging more people to subscribe to a term deposit. But, to do so we need an efficient way such that, more time can be spent on customers having more probability of accepting the subscription offer. Since, this way we can save time and money of the bank.

The workflow is as follows:

1. First step is Exploratory Data Analysis on various feature categories like Bank client data, Social and economic context attributes & Related with the last contact of the current campaign.
2. Second step is to Data Preparation using various encoding techniques.
3. In the third step, we apply Logistic Regression & KNN Model.
4. At last we compare both our results.

Models tried are:
- Logistic Regression
- KNN

Performance Metrics used are:
- Precision
- Recall
- F1-Score

In one line the objective can be stated as:

"Given all the basic information of a customer, we want to predict if they are interested in subscribing to a term deposit or not ".

Since we have to predict yes or no output, this is a "**Binary-Class Classification Problem**".

2 DATA AND FEATURES

2.1 Data

The data is related with direct marketing campaigns of a Portuguese banking institution. The marketing campaigns were based on phone calls.

Often, more than one contact to the same client was required, in order to access if the product (bank term deposit) would be ('yes') or not ('no') subscribed.

Dataset is taken from:

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

2.2 Features

In total there are 21 features, we shall define all the features in 4 categories here:

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.

Other attributes:

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 clients 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')

3 EXPLORATORY DATA ANALYSIS

3.1 Age

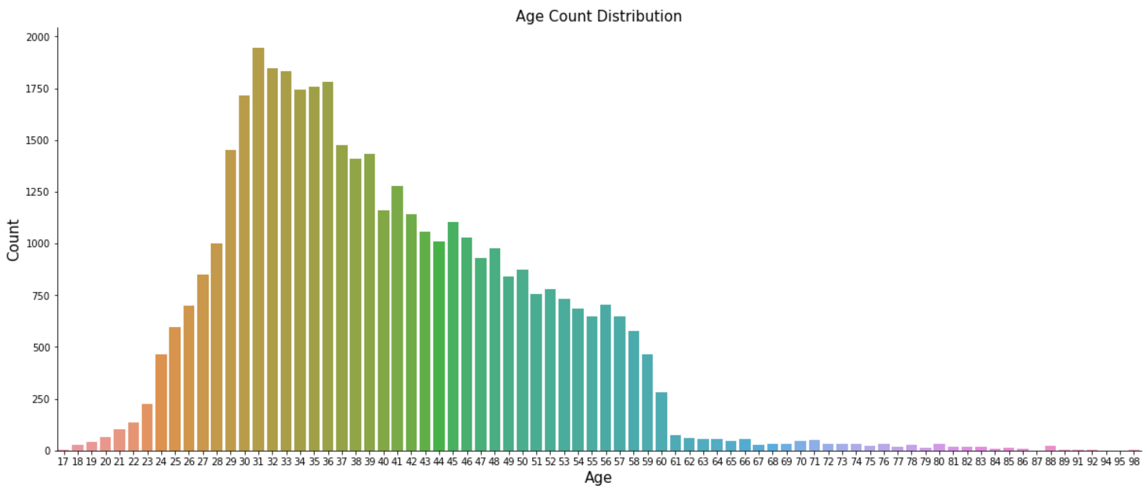


Figure 1: Histogram between Count and Age.

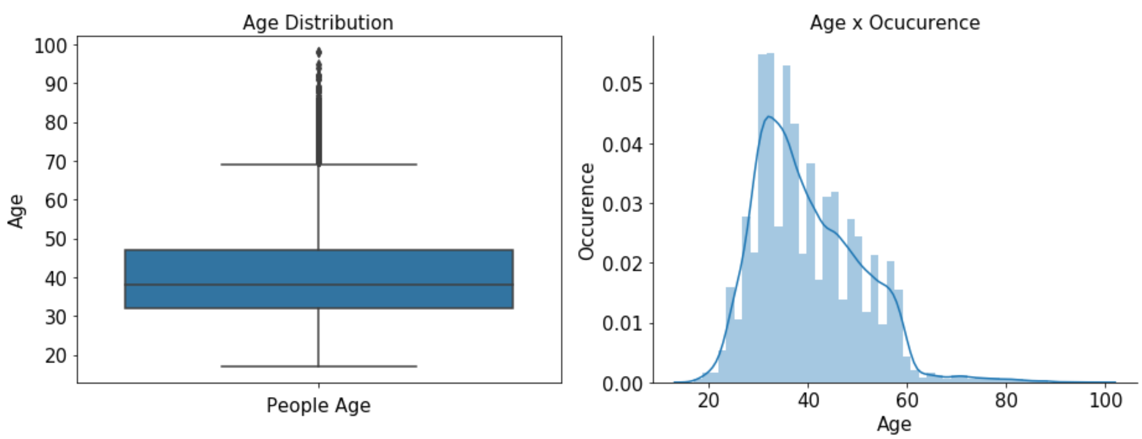


Figure 2: Box-Plot and PDF of AGE

Observation: The ages don't mean to much, has a medium dispersion and doesn't make sense relate with other variables and thus we guess it will not tell any insight in future.

Age > 69.5 are Outliers
Thus, The Number of outliers: 469

3.2 Jobs, Marital and Education

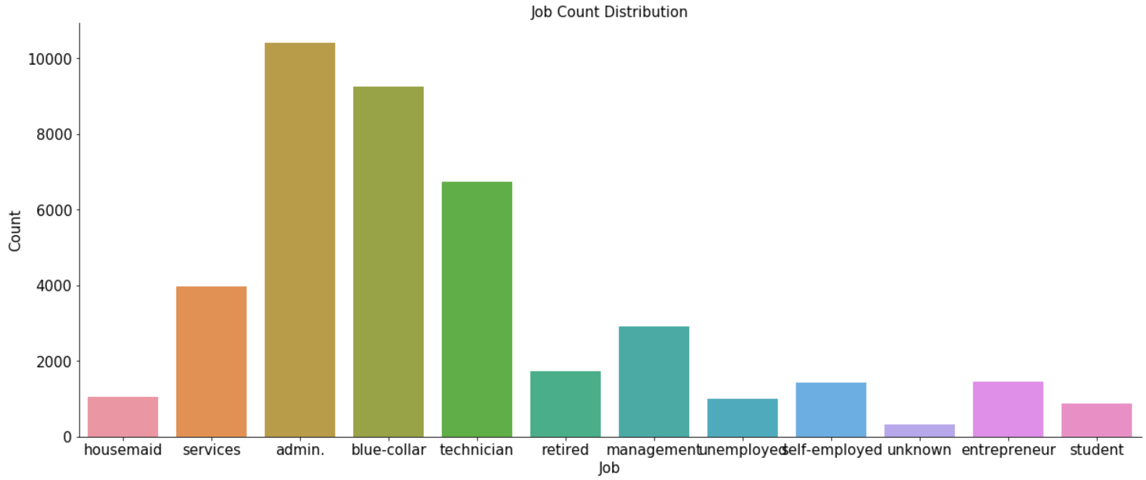


Figure 3: Histogram between Count and Job.

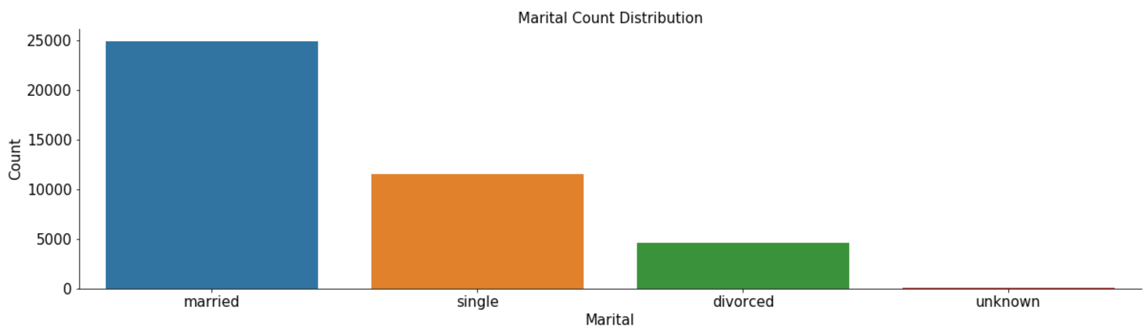


Figure 4: Histogram between Count and Marital.

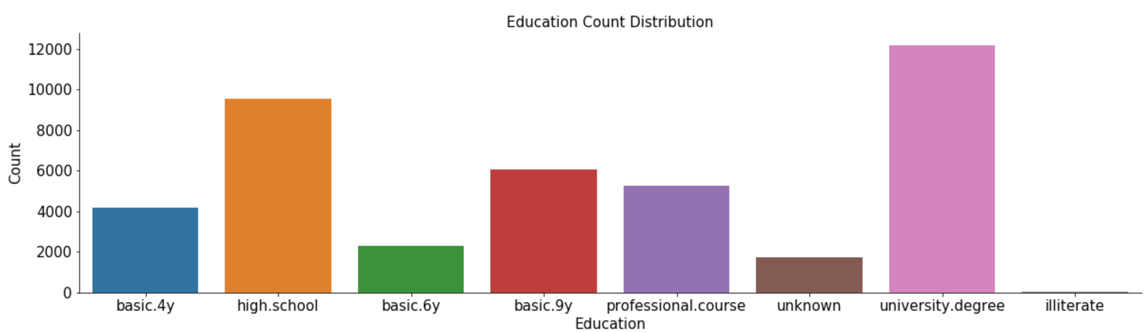


Figure 5: Histogram between Count and Education.

Observation: if we related with the other ones it is not conclusive, all this kind of variables has yes, unknown and no for loan, default and housing.

3.3 Default, Loan and Housing

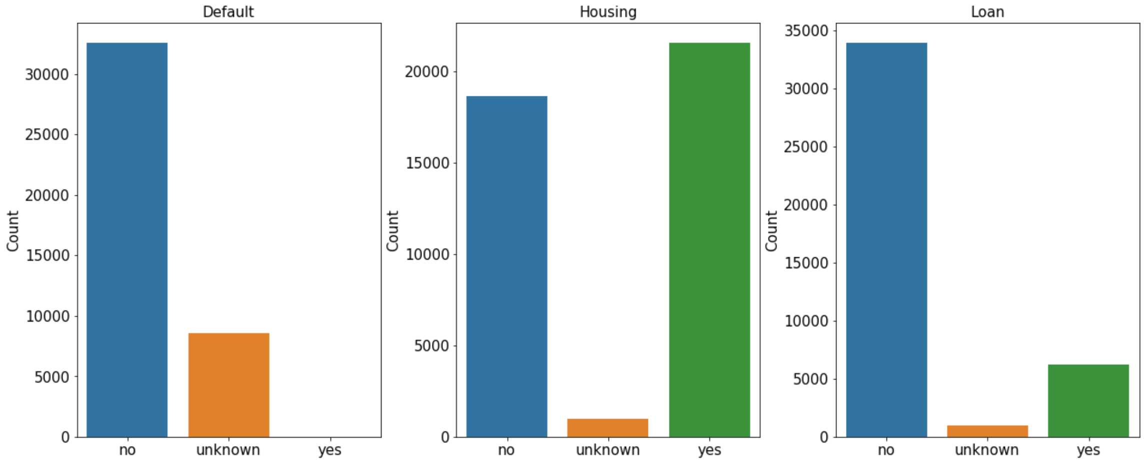


Figure 6: Histogram between Count and (Default, Housing & Loan).

Observation:

Default:
No credit in default: 32588
Unknown credit in default: 8597
Yes to credit in default: 3

Housing:
No housing in loan: 18622
Unknown housing in loan: 990
Yes to housing in loan: 21576

3.4 Durations

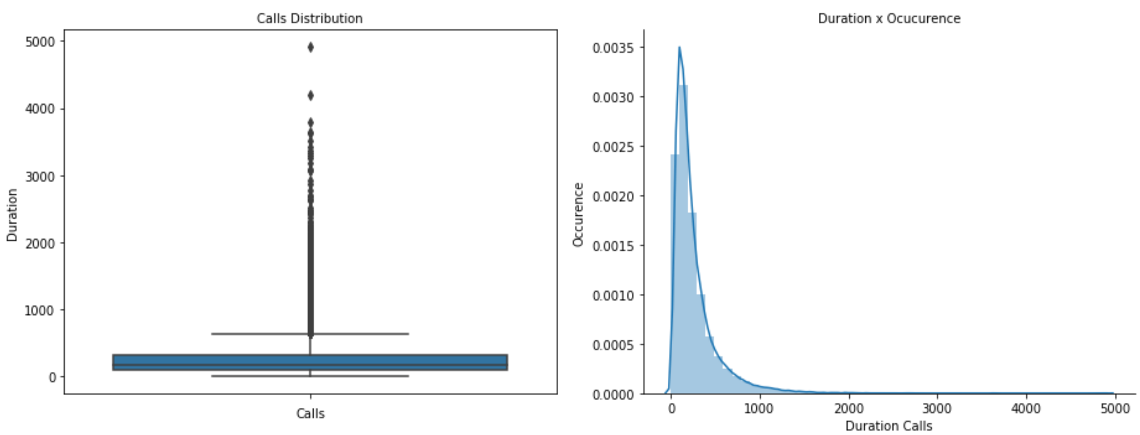


Figure 7: Box-Plot and PDF of Durations

Observation:

Duration > 644.5 are Outliers
Thus, The Number of outliers: 2963

3.5 Contact, Month & Day of Week

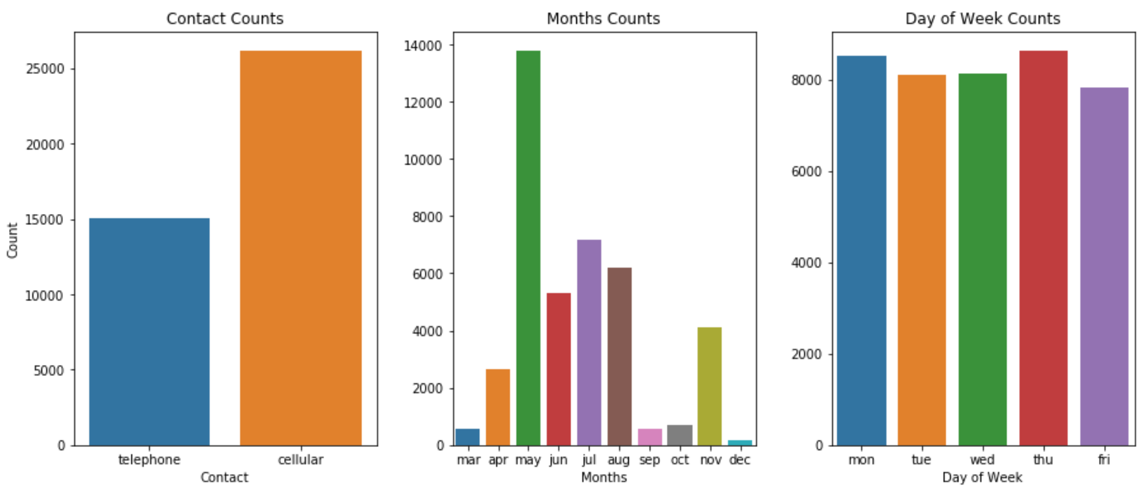


Figure 8: Histogram between Count and (Default, Housing & Loan).

4 DATA PREPARATION

Here, we hot-encoded all the categorical features:

- Job, Marital, Education, Default, Housing, Loan
- Contact, month, day of week

Also, some features were divided into segments based on limits:

- Age
- Duration

Finally, the dataset was standardized using sklearn library.

5 MODEL

5.1 Logistic Regression

```
[[6984   89]
 [ 721  206]]
90.0
```

Figure 9: Confusion Matrix & Accuracy Score when LR was applied.

5.2 K Nearest Neighbour

```
k=1 84.74 (+/- 0.75)
k=2 88.93 (+/- 0.44)
k=3 88.43 (+/- 0.38)
k=4 89.32 (+/- 0.44)
k=5 89.08 (+/- 0.43)
k=6 89.56 (+/- 0.41)
k=7 89.49 (+/- 0.38)
k=8 89.56 (+/- 0.53)
k=9 89.43 (+/- 0.51)
k=10 89.64 (+/- 0.50)
k=11 89.53 (+/- 0.48)
k=12 89.69 (+/- 0.49)
k=13 89.63 (+/- 0.50)
k=14 89.74 (+/- 0.51)
k=15 89.69 (+/- 0.45)
k=16 89.79 (+/- 0.44)
k=17 89.71 (+/- 0.45)
k=18 89.81 (+/- 0.45)
k=19 89.81 (+/- 0.48)
k=20 89.75 (+/- 0.54)
k=21 89.78 (+/- 0.54)
k=22 89.80 (+/- 0.50)
k=23 89.77 (+/- 0.55)
k=24 89.74 (+/- 0.49)
k=25 89.74 (+/- 0.48)
The optimal number of neighbors is 17 with 89.8%
```

Figure 10: Calculating best suitable ‘k’

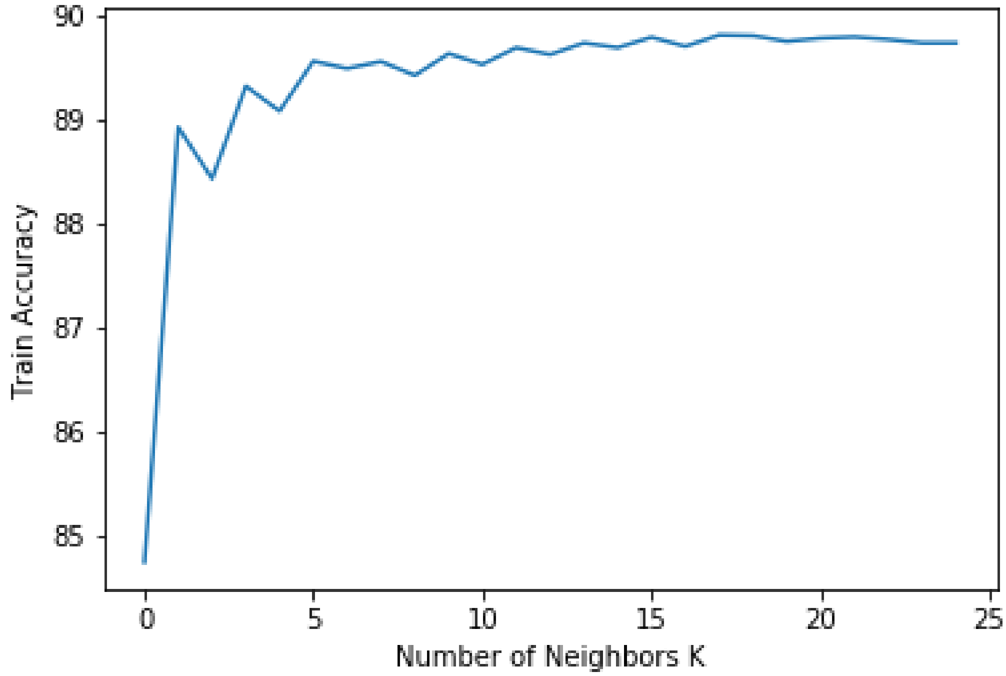


Figure 11: Calculating best suitable 'k'

[[6967 106]
[712 215]]
90.0

Figure 12: Confusion Matrix & Accuracy Score when KNN was applied.

6 RESULTS AND CONCLUSIONS

6.1 Comparisons

Out[97]:

	Models	Score
0	Logistic Model	0.900175
1	K-Near Neighbors	0.899000

Table 1: Depicting Comparison between both models. Both gives similar outputs

So now we have to decide which one is the best model, and we have two types of wrong values:

1. False Positive, means the client did NOT SUBSCRIBED to term deposit, but the model thinks he did.
2. False Negative, means the client SUBSCRIBED to term deposit, but the model said he did not.

The first one its most harmful, because we already have that client but we don't and maybe we lost him in other future campaigns

The second is not good but its ok, we have that client and in the future we'll discovery that in truth he's already our client

So, our objective here, is to find the best model by ROC Curve with the lowest False Positive as possible.

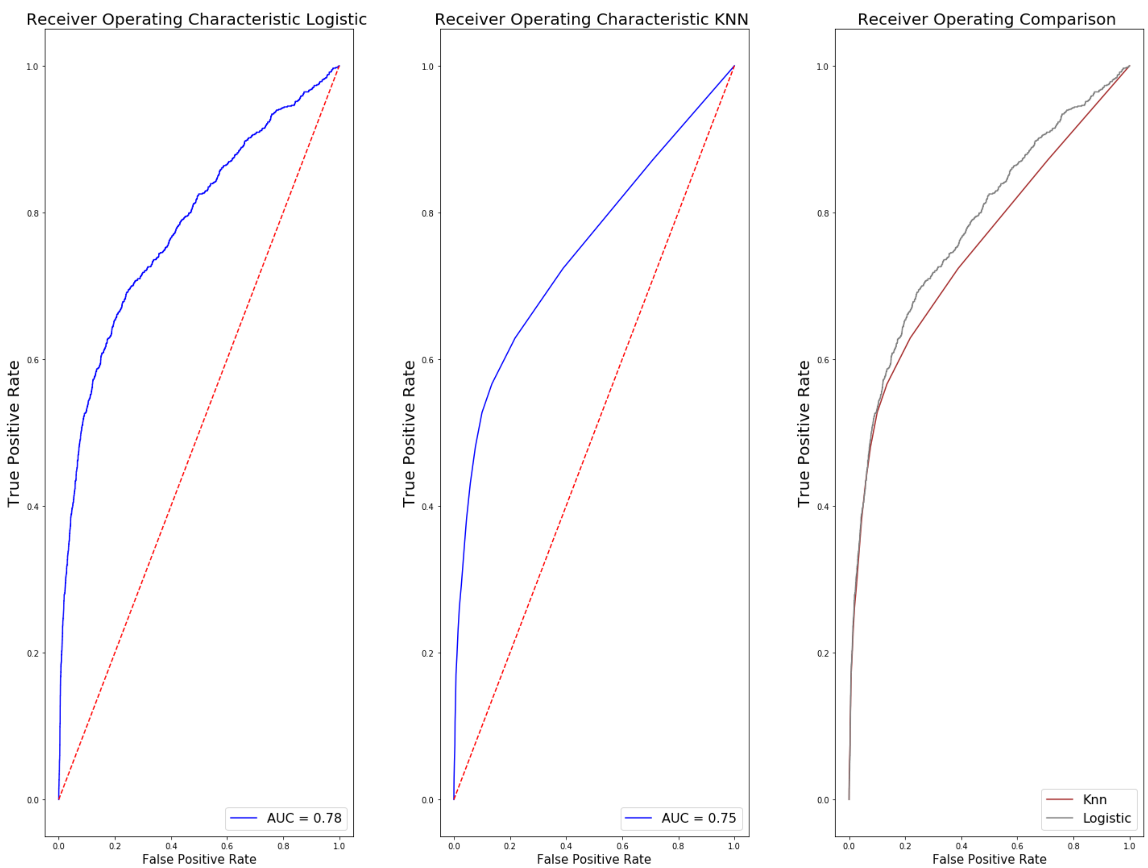


Figure 13: The ROC of both the models. An area of 1 represents a perfect test; an area of .5 represents a worthless test. Here We Can see KNN beats LR model. Thus, we have chosen KNN as our final model.

6.2 Choosing KNN

KNN Reports

	precision	recall	f1-score	support
0	0.91	0.99	0.94	7073
1	0.67	0.23	0.34	927
avg / total	0.88	0.90	0.88	8000

Figure 14: The complete report of KNN model thus implemented.

6.3 Conclusions

- Using LR was efficient but it got beaten by KNN model in terms of False Positive value.

- Applying KNN on the dataset gave us following measures:

➢ Precision Score = 0.88
➢ Recall Score = 0.9
➢ F1-score = 0.88

Thus, we shall use KNN model for future predictions of eligible customers, whom the bank should pursue regarding their term deposit related advertisements.

7 CODE

Bank Marketing

by Chinmay Sathe

Dataset from : http://archive.ics.uci.edu/ml/datasets/Bank+Marketing# (http://archive.ics.uci.edu/ml/datasets/Bank+Marketing)

1. Reading Data

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
import warnings
warnings.filterwarnings('ignore')
```

+ Code + Markdown

[2]:

```
bank = pd.read_csv('../input/bank-additional-full.csv', sep = ';')
#Creating dummy variable for yes/no
y = pd.get_dummies(bank['y'], columns = ['y'], prefix = ['y'], drop_first = True)
bank.head()
```

Out[2]:

	age	job	marital	education	default	housing	loan	contact	month	day_of_week	duration	campaign	pdays	previous	poutcome	emp.va
0	56	housemaid	married	basic.4y	no	no	no	telephone	may	mon	261	1	999	0	nonexistent	
1	57	services	married	high.school	unknown	no	no	telephone	may	mon	149	1	999	0	nonexistent	
2	37	services	married	high.school	no	yes	no	telephone	may	mon	226	1	999	0	nonexistent	
3	40	admin.	married	basic.6y	no	no	no	telephone	may	mon	151	1	999	0	nonexistent	
4	56	services	married	high.school	no	no	yes	telephone	may	mon	307	1	999	0	nonexistent	

[3]:

```
bank.isnull().any()
```

Out[3]:

```
age               False
job               False
marital           False
education         False
default           False
housing           False
loan              False
contact           False
month             False
day_of_week       False
duration          False
campaign          False
pdays             False
previous          False
poutcome          False
emp.var.rate      False
cons.price.idx    False
cons.conf.idx     False
euribor3m         False
nr.employed       False
y                 False
dtype: bool
```

[4]:

```
bank.columns
```

Out[4]:

```
Index(['age', 'job', 'marital', 'education', 'default', 'housing', 'loan',
       'contact', 'month', 'day_of_week', 'duration', 'campaign', 'pdays',
       'previous', 'poutcome', 'emp.var.rate', 'cons.price.idx',
       'cons.conf.idx', 'euribor3m', 'nr.employed', 'y'],
      dtype='object')
```

2. Exploratory Data Analaysis

2.1 Bank client data

[5]:

```
#Taking only client data into consideration, for ease of analysis. This has no effect on
#time complexity of the analysis process.
#Taking only age,job,marital,education,default,housing,loan of all features.
bank_client = bank.iloc[: , 0:7]
bank_client.head()
```

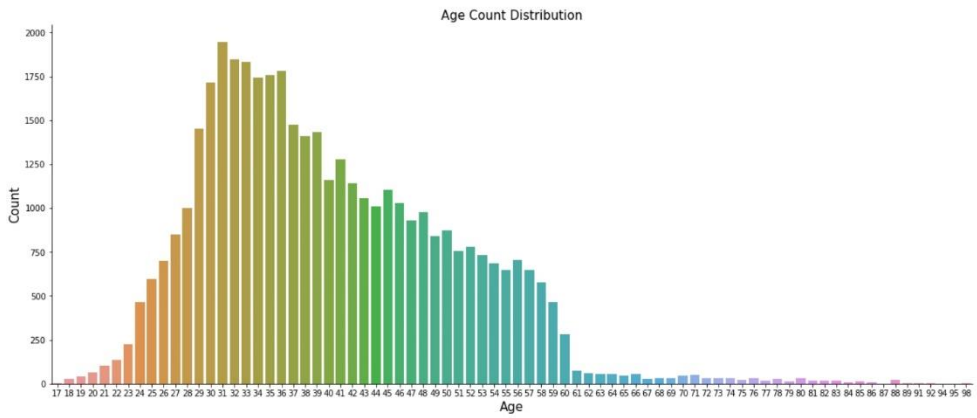
Out[5]:

	age	job	marital	education	default	housing	loan
0	56	housemaid	married	basic.4y	no	no	no
1	57	services	married	high.school	unknown	no	no
2	37	services	married	high.school	no	yes	no
3	40	admin.	married	basic.6y	no	no	no
4	56	services	married	high.school	no	no	yes

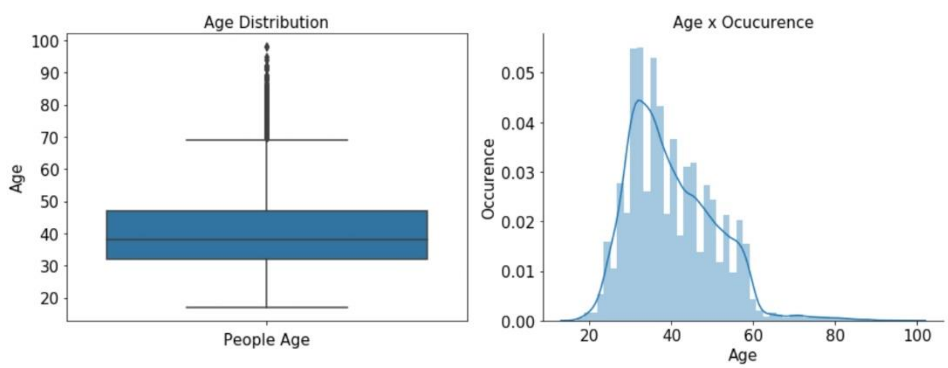
2.1.1 Age

+ Code + Markdown

[9]:
```
fig, ax = plt.subplots()
fig.set_size_inches(20, 8)
sns.countplot(x = 'age', data = bank_client)
ax.set_xlabel('Age', fontsize=15)
ax.set_ylabel('Count', fontsize=15)
ax.set_title('Age Count Distribution', fontsize=15)
sns.despine()
```



[10]:
```
fig, (ax1, ax2) = plt.subplots(nrows = 1, ncols = 2, figsize = (13, 5))
sns.boxplot(x = 'age', data = bank_client, orient = 'v', ax = ax1)
ax1.set_xlabel('People Age', fontsize=15)
ax1.set_ylabel('Age', fontsize=15)
ax1.set_title('Age Distribution', fontsize=15)
ax1.tick_params(labelsize=15)

sns.distplot(bank_client['age'], ax = ax2)
sns.despine(ax = ax2)
ax2.set_xlabel('Age', fontsize=15)
ax2.set_ylabel('Occurence', fontsize=15)
ax2.set_title('Age x Ocucurence', fontsize=15)
ax2.tick_params(labelsize=15)

plt.subplots_adjust(wspace=0.5)
plt.tight_layout()
```



[36]:

```
# here Quantiles[0.25]= 25% quantile & Quantiles[1.00]= 100% quantile
Quantiles = bank_client['age'].quantile(q = [.25,.5,.75,1])
limitedAge = Quantiles[0.75] + 1.5*(Quantiles[0.75] - Quantiles[0.25])
print('Age > ',limitedAge,' are Outliers ')
print('Thus, The Number of outliers: ', bank_client[bank_client['age'] > limitedAge]['age'].count())
```

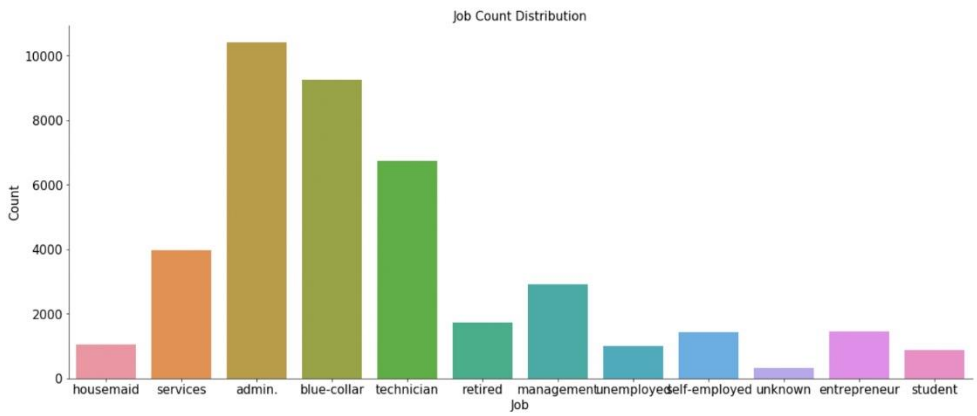
```
Age >  69.5  are Outliers
Thus, The Number of outliers:  469
```

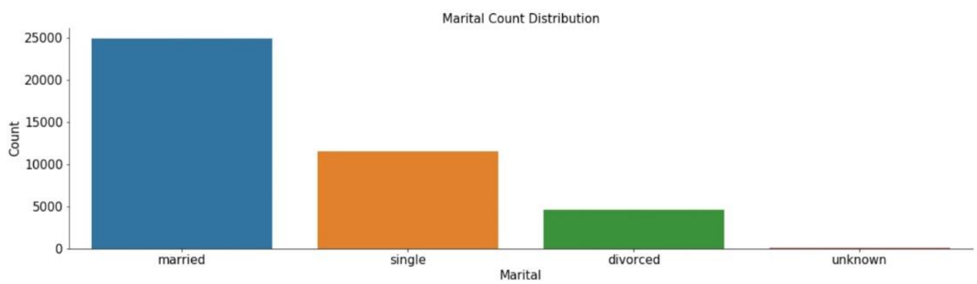
2.1.2 JOBS,MARITAL,EDUCATION

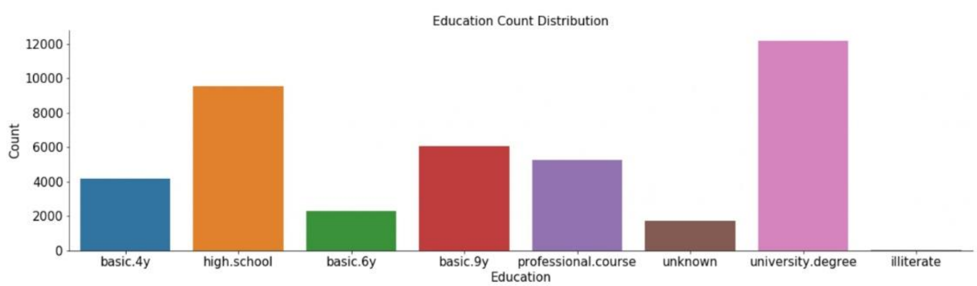
+ Code + Markdown

[50]:

```python
def featu_count_histo(height,width,x,label):
    title = label+' Count Distribution'
    fig, ax = plt.subplots()
    fig.set_size_inches(height,width)
    sns.countplot(x = x, data = bank_client)
    ax.set_xlabel(label, fontsize=15)
    ax.set_ylabel('Count', fontsize=15)
    ax.set_title(title, fontsize=15)
    ax.tick_params(labelsize=15)
    sns.despine()

inchh =[20,20,20]
inchw =[8,5,5]
feat = ['job','marital','education']
Labelx = ['Job','Marital','Education']

for x in zip(inchh,inchw,feat,Labelx):
    featu_count_histo(x[0],x[1],x[2],x[3])
```





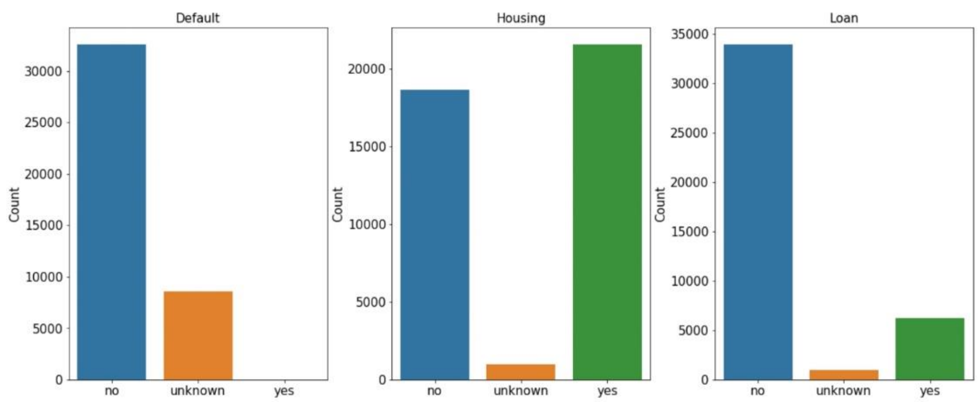


2.1.3 DEFAULT, HOUSING, LOAN

+ Code   + Markdown

[57]:

```
fig, (ax1, ax2, ax3) = plt.subplots(nrows = 1, ncols = 3, figsize = (20,8))
sns.countplot(x = 'default', data = bank_client, ax = ax1, order = ['no', 'unknown', 'yes'])
ax1.set_title('Default', fontsize=15)
ax1.set_xlabel('')
ax1.set_ylabel('Count', fontsize=15)
ax1.tick_params(labelsize=15)

sns.countplot(x = 'housing', data = bank_client, ax = ax2, order = ['no', 'unknown', 'yes'])
ax2.set_title('Housing', fontsize=15)
ax2.set_xlabel('')
ax2.set_ylabel('Count', fontsize=15)
ax2.tick_params(labelsize=15)

sns.countplot(x = 'loan', data = bank_client, ax = ax3, order = ['no', 'unknown', 'yes'])
ax3.set_title('Loan', fontsize=15)
ax3.set_xlabel('')
ax3.set_ylabel('Count', fontsize=15)
ax3.tick_params(labelsize=15)

plt.subplots_adjust(wspace=0.25)
```



[58]:

```
print('Default:\n No credit in default:'     , bank_client[bank_client['default'] == 'no']     ['age'].count(),
              '\n Unknown credit in default:', bank_client[bank_client['default'] == 'unknown']['age'].count(),
              '\n Yes to credit in default:' , bank_client[bank_client['default'] == 'yes']    ['age'].count())
```

```
Default:
 No credit in default: 32588
 Unknown credit in default: 8597
 Yes to credit in default: 3
```

[59]:

```
print('Housing:\n No housing in loan:'     , bank_client[bank_client['housing'] == 'no']     ['age'].count(),
              '\n Unknown housing in loan:', bank_client[bank_client['housing'] == 'unknown']['age'].count(),
              '\n Yes to housing in loan:' , bank_client[bank_client['housing'] == 'yes']    ['age'].count())
```

```
Housing:
 No housing in loan: 18622
 Unknown housing in loan: 990
 Yes to housing in loan: 21576
```

[60]:
```
print('Housing:\n No to personal loan:'      , bank_client[bank_client['loan'] == 'no']     ['age'].count(),
              '\n Unknown to personal loan:', bank_client[bank_client['loan'] == 'unknown']['age'].count(),
              '\n Yes to personal loan:'     , bank_client[bank_client['loan'] == 'yes']    ['age'].count())
```

```
Housing:
 No to personal loan: 33950
 Unknown to personal loan: 990
 Yes to personal loan: 6248
```

2.2 Related with the last contact of the current campaign

[66]:
```
#Again Data Slicing for each of view
bank_related = bank.iloc[: , 7:11]
bank_related.head()
```

Out[66]:

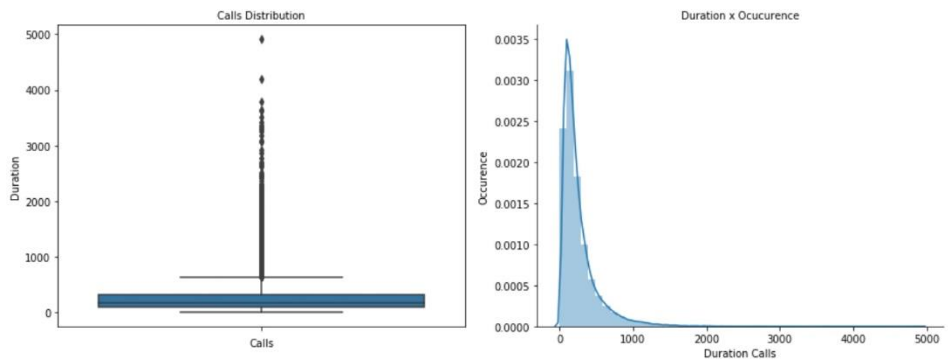
	contact	month	day_of_week	duration
0	telephone	may	mon	261
1	telephone	may	mon	149
2	telephone	may	mon	226
3	telephone	may	mon	151
4	telephone	may	mon	307

[67]:
```
bank_related.isnull().any()
```

Out[67]:
```
contact        False
month          False
day_of_week    False
duration       False
dtype: bool
```

2.2.1 Duration

[69]:
```
fig, (ax1, ax2) = plt.subplots(nrows = 1, ncols = 2, figsize = (13, 5))
sns.boxplot(x = 'duration', data = bank_related, orient = 'v', ax = ax1)
ax1.set_xlabel('Calls', fontsize=10)
ax1.set_ylabel('Duration', fontsize=10)
ax1.set_title('Calls Distribution', fontsize=10)
ax1.tick_params(labelsize=10)

sns.distplot(bank_related['duration'], ax = ax2)
sns.despine(ax = ax2)
ax2.set_xlabel('Duration Calls', fontsize=10)
ax2.set_ylabel('Occurence', fontsize=10)
ax2.set_title('Duration x Ocucurence', fontsize=10)
ax2.tick_params(labelsize=10)

plt.subplots_adjust(wspace=0.5)
plt.tight_layout()
```



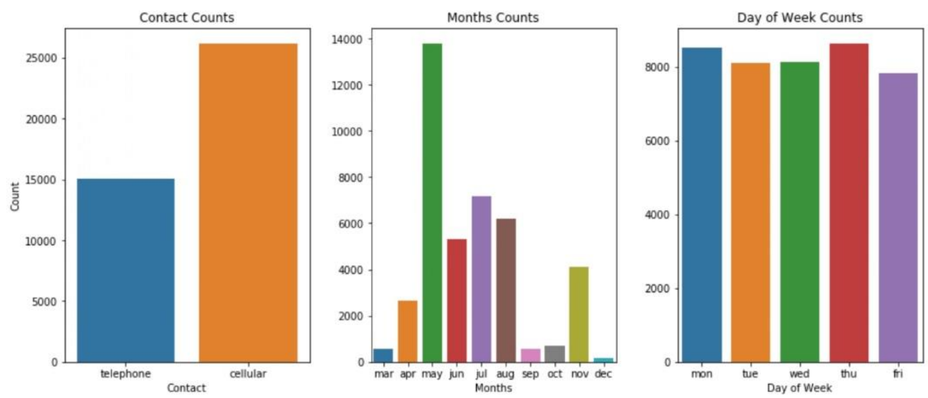
PLease note: duration is different from age, Age has 78 values and Duration has 1544 different values

[74]:
```
# here Quantiles[0.25]= 25% quantile & Quantiles[1.00]= 100% quantile
Quantiles = bank_related['duration'].quantile(q = [.25,.5,.75,1])
limitedAge = Quantiles[0.75] + 1.5*(Quantiles[0.75] - Quantiles[0.25])
print('Duration > ',limitedAge,' are Outliers ')
print('Thus, The Number of outliers: ', bank_client[bank_related['duration'] > limitedAge]['age'].count())
```

```
Duration >  644.5  are Outliers
Thus, The Number of outliers:  2963
```

2.2.2 Contact, Month, Day of Week

[76]:
```
fig, (ax1, ax2, ax3) = plt.subplots(nrows = 1, ncols = 3, figsize = (15,6))
sns.countplot(bank_related['contact'], ax = ax1)
ax1.set_xlabel('Contact', fontsize = 10)
ax1.set_ylabel('Count', fontsize = 10)
ax1.set_title('Contact Counts')
ax1.tick_params(labelsize=10)

sns.countplot(bank_related['month'], ax = ax2, order = ['mar', 'apr', 'may', 'jun', 'jul', 'aug', 'sep', 'oct', 'nov', 'dec'
ax2.set_xlabel('Months', fontsize = 10)
ax2.set_ylabel('')
ax2.set_title('Months Counts')
ax2.tick_params(labelsize=10)

sns.countplot(bank_related['day_of_week'], ax = ax3)
ax3.set_xlabel('Day of Week', fontsize = 10)
ax3.set_ylabel('')
ax3.set_title('Day of Week Counts')
ax3.tick_params(labelsize=10)

plt.subplots_adjust(wspace=0.25)
```



2.3 Social and economic context attributes

[79]:
```
#Slicing Dataset
bank_se = bank.loc[: , ['emp.var.rate', 'cons.price.idx', 'cons.conf.idx', 'euribor3m', 'nr.employed']]
bank_se.head()
```

Out[79]:

	emp.var.rate	cons.price.idx	cons.conf.idx	euribor3m	nr.employed
0	1.1	93.994	-36.4	4.857	5191.0
1	1.1	93.994	-36.4	4.857	5191.0
2	1.1	93.994	-36.4	4.857	5191.0
3	1.1	93.994	-36.4	4.857	5191.0
4	1.1	93.994	-36.4	4.857	5191.0

2.4 Other attributes

[80]:
```
bank_o = bank.loc[: , ['campaign', 'pdays','previous', 'poutcome']]
bank_o.head()
```

Out[80]:

	campaign	pdays	previous	poutcome
0	1	999	0	nonexistent
1	1	999	0	nonexistent
2	1	999	0	nonexistent
3	1	999	0	nonexistent
4	1	999	0	nonexistent

3 Data Preparation

3.1 Data Client Dataset

[86]:
```python
from sklearn.preprocessing import LabelEncoder
labelencoder_X = LabelEncoder()
bank_client['job']      = labelencoder_X.fit_transform(bank_client['job'])
bank_client['marital']  = labelencoder_X.fit_transform(bank_client['marital'])
bank_client['education']= labelencoder_X.fit_transform(bank_client['education'])
bank_client['default']  = labelencoder_X.fit_transform(bank_client['default'])
bank_client['housing']  = labelencoder_X.fit_transform(bank_client['housing'])
bank_client['loan']     = labelencoder_X.fit_transform(bank_client['loan'])

def age(dataframe):
    dataframe.loc[dataframe['age'] <= 32, 'age'] = 1
    dataframe.loc[(dataframe['age'] > 32) & (dataframe['age'] <= 47), 'age'] = 2
    dataframe.loc[(dataframe['age'] > 47) & (dataframe['age'] <= 70), 'age'] = 3
    dataframe.loc[(dataframe['age'] > 70) & (dataframe['age'] <= 98), 'age'] = 4

    return dataframe

age(bank_client);
bank_client.head()
```

Out[86]:

	age	job	marital	education	default	housing	loan
0	1	3	1	0	0	0	0
1	1	7	1	3	1	0	0
2	1	7	1	3	0	2	0
3	1	0	1	1	0	0	0
4	1	7	1	3	0	0	2

3.2 Related with the last contact of the current campaign Dataset

[88]:
```python
# Label encoder order is alphabetical
from sklearn.preprocessing import LabelEncoder
labelencoder_X = LabelEncoder()
bank_related['contact']     = labelencoder_X.fit_transform(bank_related['contact'])
bank_related['month']       = labelencoder_X.fit_transform(bank_related['month'])
bank_related['day_of_week'] = labelencoder_X.fit_transform(bank_related['day_of_week'])
def duration(data):

    data.loc[data['duration'] <= 102, 'duration'] = 1
    data.loc[(data['duration'] > 102) & (data['duration'] <= 180)  , 'duration']    = 2
    data.loc[(data['duration'] > 180) & (data['duration'] <= 319)  , 'duration']   = 3
    data.loc[(data['duration'] > 319) & (data['duration'] <= 644.5), 'duration'] = 4
    data.loc[data['duration']  > 644.5, 'duration'] = 5

    return data
duration(bank_related);
bank_related.head()
```

Out[88]:

	contact	month	day_of_week	duration
0	1	6	1	1
1	1	6	1	1
2	1	6	1	1
3	1	6	1	1
4	1	6	1	1

3.3 Final Dataset

```
[90]:
bank_final= pd.concat([bank_client, bank_related, bank_se, bank_o], axis = 1)
bank_final = bank_final[['age', 'job', 'marital', 'education', 'default', 'housing', 'loan',
                     'contact', 'month', 'day_of_week', 'duration', 'emp.var.rate', 'cons.price.idx',
                     'cons.conf.idx', 'euribor3m', 'nr.employed', 'campaign', 'pdays', 'previous', 'poutcome']]
bank_final.shape
```

```
Out[90]:(41188, 20)
```

```
[91]:
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(bank_final, y, test_size = 0.1942313295, random_state = 101)

from sklearn.model_selection import KFold
from sklearn.model_selection import cross_val_score
from sklearn.metrics import confusion_matrix, accuracy_score
k_fold = KFold(n_splits=10, shuffle=True, random_state=0)
```

```
[92]:
X_train.head()
```

Out[92]:

	age	job	marital	education	default	housing	loan	contact	month	day_of_week	duration	emp.var.rate	cons.price.idx	cons.conf.idx	euribor3m
38912	1	5	1	6	0	2	0	0	7	4	1	-3.4	92.649	-30.1	0.716
9455	1	7	1	5	1	0	0	1	4	0	1	1.4	94.465	-41.8	4.967
14153	1	4	1	6	0	2	0	0	3	1	1	1.4	93.918	-42.7	4.962
25021	1	6	1	6	0	2	0	0	7	3	1	-0.1	93.200	-42.0	4.153
30911	1	5	0	0	0	2	2	0	6	3	1	-1.8	92.893	-46.2	1.344

```
[93]:
from sklearn.preprocessing import StandardScaler
sc_X = StandardScaler()
X_train = sc_X.fit_transform(X_train)
X_test = sc_X.transform(X_test)
```

4 Model

4.1 Logistic Regression

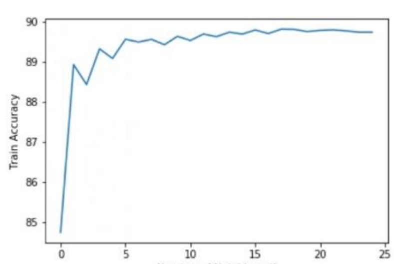
```
[94]:
from sklearn.linear_model import LogisticRegression
logmodel = LogisticRegression()
logmodel.fit(X_train,y_train)
logpred = logmodel.predict(X_test)


print(confusion_matrix(y_test, logpred))
print(round(accuracy_score(y_test, logpred),2)*100)
LOGCV = (cross_val_score(logmodel, X_train, y_train, cv=k_fold, n_jobs=1, scoring = 'accuracy').mean())
```

```
[[6984   89]
 [ 721  206]]
90.0
```

[95]:

```
from sklearn import model_selection
from sklearn.neighbors import KNeighborsClassifier

X_trainK, X_testK, y_trainK, y_testK = train_test_split(bank_final, y, test_size = 0.2, random_state = 101)

#Neighbors
neighbors = np.arange(0,25)

#Create empty list that will hold cv scores
cv_scores = []

#Perform 10-fold cross validation on training set for odd values of k:
for k in neighbors:
    k_value = k+1
    knn = KNeighborsClassifier(n_neighbors = k_value, weights='uniform', p=2, metric='euclidean')
    kfold = model_selection.KFold(n_splits=10, random_state=123)
    scores = model_selection.cross_val_score(knn, X_trainK, y_trainK, cv=kfold, scoring='accuracy')
    cv_scores.append(scores.mean()*100)
    print("k=%d %0.2f (+/- %0.2f)" % (k_value, scores.mean()*100, scores.std()*100))
optimal_k = neighbors[cv_scores.index(max(cv_scores))]
print ("The optimal number of neighbors is %d with %0.1f%%" % (optimal_k, cv_scores[optimal_k]))

plt.plot(neighbors, cv_scores)
plt.xlabel('Number of Neighbors K')
plt.ylabel('Train Accuracy')
plt.show()
```

```
k=1 84.74 (+/- 0.75)
k=2 88.93 (+/- 0.44)
k=3 88.43 (+/- 0.38)
k=4 89.32 (+/- 0.44)
k=5 89.08 (+/- 0.43)
k=6 89.56 (+/- 0.41)
k=7 89.49 (+/- 0.38)
k=8 89.56 (+/- 0.53)
k=9 89.43 (+/- 0.51)
k=10 89.64 (+/- 0.50)
k=11 89.53 (+/- 0.48)
k=12 89.69 (+/- 0.49)
k=13 89.63 (+/- 0.50)
k=14 89.74 (+/- 0.51)
k=15 89.69 (+/- 0.45)
k=16 89.79 (+/- 0.44)
k=17 89.71 (+/- 0.45)
k=18 89.81 (+/- 0.45)
k=19 89.81 (+/- 0.48)
k=20 89.75 (+/- 0.54)
k=21 89.78 (+/- 0.54)
k=22 89.80 (+/- 0.50)
k=23 89.77 (+/- 0.55)
k=24 89.74 (+/- 0.49)
k=25 89.74 (+/- 0.48)
The optimal number of neighbors is 17 with 89.8%
```



[96]:

```
from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier(n_neighbors=22)
knn.fit(X_train, y_train)
knnpred = knn.predict(X_test)

print(confusion_matrix(y_test, knnpred))
print(round(accuracy_score(y_test, knnpred),2)*100)
KNNCV = (cross_val_score(knn, X_train, y_train, cv=k_fold, n_jobs=1, scoring = 'accuracy').mean())
```

```
[[6967  106]
 [ 712  215]]
90.0
```

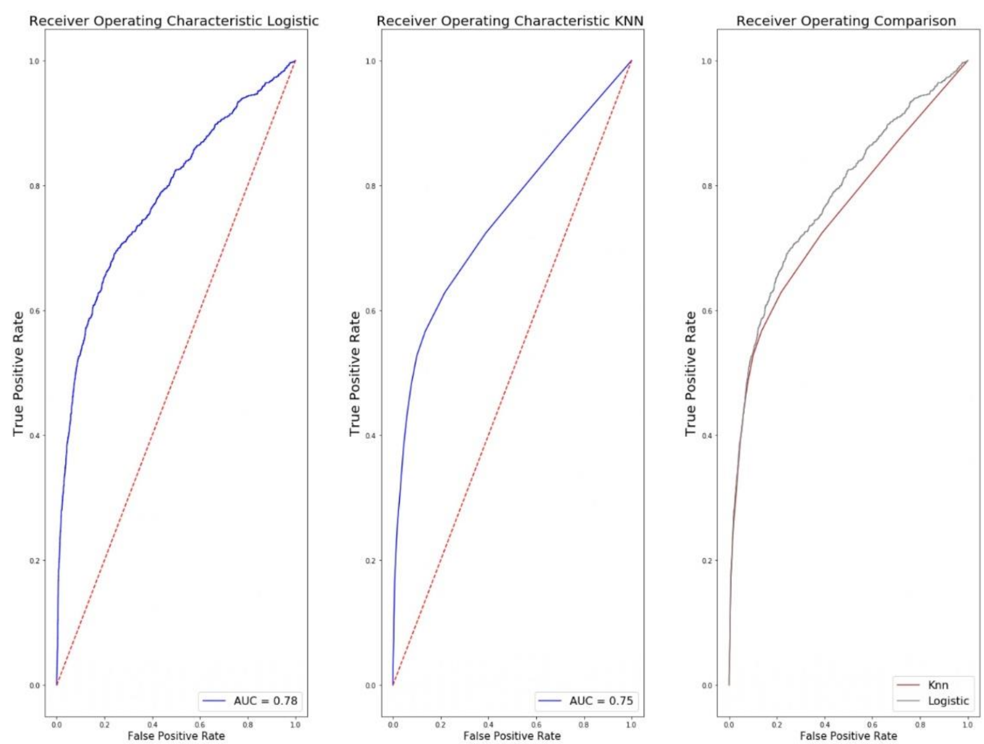
[97]:

```
models = pd.DataFrame({
                'Models': ['Logistic Model','K-Near Neighbors'],
                'Score':  [LOGCV,KNNCV]})

models.sort_values(by='Score', ascending=False)
```

Out[97]:

	Models	Score
0	Logistic Model	0.900175
1	K-Near Neighbors	0.899000

[99]:
```
from sklearn import metrics
fig, (ax1, ax2,ax3) = plt.subplots(nrows = 1, ncols = 3, figsize = (20,15))

#Logistic Regression
probs = logmodel.predict_proba(X_test)
preds = probs[:,1]
fprlog, tprlog, thresholdlog = metrics.roc_curve(y_test, preds)
roc_auclog = metrics.auc(fprlog, tprlog)

ax1.plot(fprlog, tprlog, 'b', label = 'AUC = %0.2f' % roc_auclog)
ax1.plot([0, 1], [0, 1],'r--')
ax1.set_title('Receiver Operating Characteristic Logistic ',fontsize=20)
ax1.set_ylabel('True Positive Rate',fontsize=20)
ax1.set_xlabel('False Positive Rate',fontsize=15)
ax1.legend(loc = 'lower right', prop={'size': 16})

#Knn
probs = knn.predict_proba(X_test)
preds = probs[:,1]
fprknn, tprknn, thresholdknn = metrics.roc_curve(y_test, preds)
roc_aucknn = metrics.auc(fprknn, tprknn)

ax2.plot(fprknn, tprknn, 'b', label = 'AUC = %0.2f' % roc_aucknn)
ax2.plot([0, 1], [0, 1],'r--')
ax2.set_title('Receiver Operating Characteristic KNN ',fontsize=20)
ax2.set_ylabel('True Positive Rate',fontsize=20)
ax2.set_xlabel('False Positive Rate',fontsize=15)
ax2.legend(loc = 'lower right', prop={'size': 16})

#ALL PLOTS ----------------------------------

ax3.plot(fprknn, tprknn, 'b', label = 'Knn', color='brown')
ax3.plot(fprlog, tprlog, 'b', label = 'Logistic', color='grey')
ax3.set_title('Receiver Operating Comparison ',fontsize=20)
ax3.set_ylabel('True Positive Rate',fontsize=20)
ax3.set_xlabel('False Positive Rate',fontsize=15)
ax3.legend(loc = 'lower right', prop={'size': 16})
plt.subplots_adjust(wspace=0.2)
plt.tight_layout()
```



[103]:
```
#We Chose KNN based on above graphs
from sklearn.metrics import classification_report
print('KNN Reports\n',classification_report(y_test, knnpred))
```

```
KNN Reports
              precision    recall  f1-score   support

          0       0.91      0.99      0.94      7073
          1       0.67      0.23      0.34       927

avg / total       0.88      0.90      0.88      8000
```

[125]:
```
#Recall Score
print(round(metrics.recall_score(y_test, knnpred),2))
```

```
0.23
```

[126]:
```
#Precision Score
print(round(metrics.precision_score(y_test, knnpred),2))
```

```
0.67
```

8 REFERENCES

https://www.brainkart.com/article/Bank-Marketing_6027/

https://www.omicsonline.org/open-access/marketing-of-financial-and-banking-products-an-example-frombangladeshi-bank-2168-9601-1000159.php?aid=76106