**Logistic Regression**

Logistic regression is a statistical method for predicting binary classes. The outcome or target variable is dichotomous in nature. Dichotomous means there are only two possible classes. For example, it can be used for cancer detection problems. It computes the probability of an event occurrence. It is a special case of linear regression where the target variable is categorical in nature. It uses a log of odds as the dependent variable. Logistic Regression predicts the probability of occurrence of a binary event utilizing a logit function.

**Linear Regression Equation:**

****

**Sigmoid Function**

****

**Apply Sigmoid function on linear regression:**

****

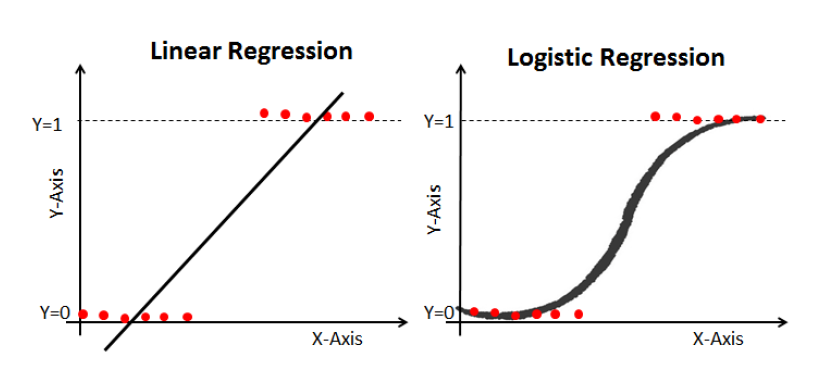
The last equation is called the Logistic Equation which is

responsible for the calculation of Logistic Regression

**Linear Regression Vs. Logistic Regression**

Linear regression gives you a continuous output, but logistic regression provides a constant output. An example of the continuous output is house price and stock price. Example's of the discrete output is predicting whether a patient has cancer or not, predicting whether the customer will churn. Linear

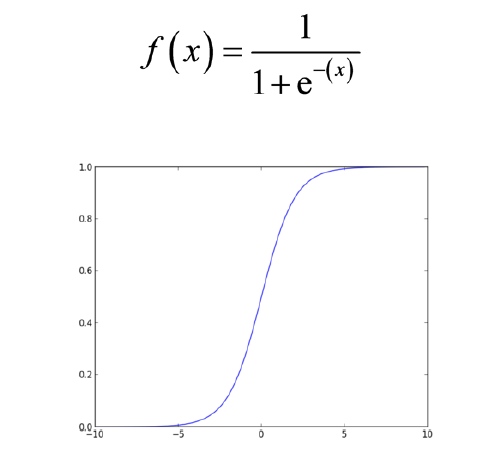
regression is estimated using Ordinary Least Squares (OLS) while logistic regression is estimated using Maximum Likelihood Estimation (MLE) approach.



**Sigmoid Function**

The sigmoid function, also called logistic function gives an ‘S’ shaped curve that can take any real-valued number and map it into a value between 0 and 1. If the curve goes to positive infinity, y predicted will become 1, and if the curve goes to negative infinity, y predicted will become 0. If the output of the

sigmoid function is more than 0.5, we can classify the outcome as 1 or YES, and if it is less than 0.5, we can classify it as 0 or NO. The output cannot For example: If the output is 0.75, we can say in terms of probability as: There is a 75 percent chance that patient will suffer from cancer.



In [3]: #Importing Required Packages

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)

In [6]: #Import and Show the Datasheet

data = pd.read\_csv('D:/As a Trainer/Freelance Training/ML-NIVT-

2/Logistic Regression/bank.csv', header=0)

#data = data.dropna()

print(data.shape)

print(list(data.columns))

In [7]: data.head()

In [8]: data['education'].unique()

(41188, 21)

['age', 'job', 'marital', 'education', 'default', 'housing',

'loan', 'contact', 'month', 'day\_of\_week', 'duration', 'campa

ign', 'pdays', 'previous', 'poutcome', 'emp\_var\_rate', 'cons\_

price\_idx', 'cons\_conf\_idx', 'euribor3m', 'nr\_employed', 'y']

Out[7]:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | **age** | **job** | **marital** | **education** | **default** |  |
| 0 | 44 | blue-collar | married | basic.4y | unknown | yes |
| 1 | 53 | technician | married | unknown | no | no |
| 2 | 28 | management | single | university.degree | no | yes |
| 3 | 39 | services | married | high.school | no | no |
| 4 | 55 | retired | married | basic.4y | no | yes |

**5 rows × 21 columns**

In [8]: data['education'].unique()

Out[8]: array(['basic.4y', 'unknown', 'university.degree', 'high.scho

ol',

'basic.9y', 'professional.course', 'basic.6y', 'illite

rate'],

dtype=object)

In [9]: #Let us group "basic.4y", "basic.9y" and "basic.6y" together and

call them "basic".

data['education']=np.where(data['education'] =='basic.9y', 'Basi

c', data['education'])

data['education']=np.where(data['education'] =='basic.6y', 'Basi

c', data['education'])

data['education']=np.where(data['education'] =='basic.4y', 'Basi

c', data['education'])

In [10]: data['education'].unique()

Out[10]: array(['Basic', 'unknown', 'university.degree', 'high.school',

'professional.course', 'illiterate'], dtype=object)

In [11]: data['y'].value\_counts()

Out[11]: 0 36548

1 4640

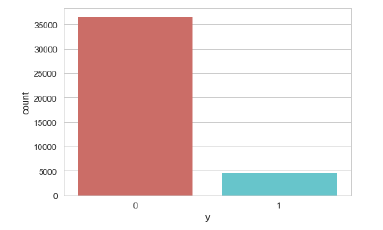
Name: y, dtype: int64

In [12]: #Using countplot to plot the count of '0' & '1'

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

plt.show()

plt.savefig('count\_plot')



<Figure size 432x288 with 0 Axes>

In [13]: #Grouping the category

data.groupby('y').mean()

Out[13]:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | age | duration | campaign | pdays | previous |
| y |  |  |  |  |  |
| 0 | 39.911185 | 220.844807 | 2.633085 | 984.113878 | 0.132374 |
| 1 | 40.913147 | 553.191164 | 2.051724 | 792.035560 | 0.492672 |

In [14]: #Goupby by Job category

data.groupby('job').mean()

Out[14]:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **age** | **duration** | **campaign** | **pday** |
| job |  |  |  |  |
| admin. | 38.187296 | 254.312128 | 2.623489 | 954.31922 |
| blue-collar | 39.555760 | 264.542360 | 2.558461 | 985.16036 |
| entrepreneur | 41.723214 | 263.267857 | 2.535714 | 981.26717 |
| housemaid | 45.500000 | 250.454717 | 2.639623 | 960.57924 |
| management | 42.362859 | 257.058140 | 2.476060 | 962.64705 |
| retired | 62.027326 | 273.712209 | 2.476744 | 897.93604 |
| Self employed | 39.949331 | 264.142153 | 2.660802 | 976.62139 |
| services | 37.926430 | 258.398085 | 2.587805 | 979.97404 |
| student | 25.894857 | 283.683429 | 2.104000 | 840.21714 |
| technician | 38.507638 | 250.232241 | 2.577339 | 964.40812 |
| unemployed | 39.733728 | 249.451677 | 2.564103 | 935.31656 |
| unknown | 45.563636 | 239.675758 | 2.648485 | 938.72727 |

In [15]: #Group by Marital

data.groupby('marital').mean()

Out[15]:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | age | duration | campaign | pdays |  |
| marital |  |  |  |  |  |
| divorced | 44.899393 | 253.790330 | 2.61340 | 968.639853 |  |
| married | 42.307165 | 257.438623 | 2.57281 | 967.247673 |  |
| single | 33.158714 | 261.524378 | 2.53380 | 949.909578 |  |
| unknown | 40.275000 | 312.725000 | 3.18750 | 937.100000 |  |

In [16]: #Goupby Education

data.groupby('education').mean()

Out[16]:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | age | duration | campaign |  |
| education |  |  |  |  |
| Basic | 42.163910 | 263.043874 | 2.559498 |  |
| high.school | 37.998213 | 260.886810 | 2.568576 |  |
| illiterate | 48.500000 | 276.777778 | 2.277778 |  |
| professional.course | 40.080107 | 252.533855 | 2.586115 |  |
| university.degree | 38.879191 | 253.223373 | 2.563527 |  |
| unknown | 43.481225 | 262.390526 | 2.596187 |  |

In [17]: # Ploting Initial Values

# The frequency of purchase of the deposit depends a great deal

on the job title.

# Thus, the job title can be a good predictor of the outcome var

iable.

# SELF

# print(data.y)

%matplotlib inline

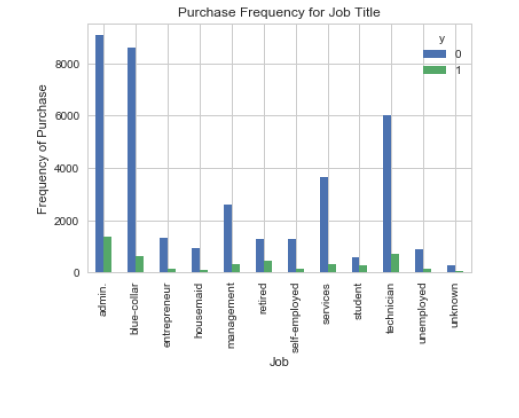
pd.crosstab(data.job,data.y).plot(kind='bar')

plt.title('Purchase Frequency for Job Title')

plt.xlabel('Job')

plt.ylabel('Frequency of Purchase')

plt.savefig('purchase\_fre\_job')



In [43]: #Stacked BAR CHART of Marital Status vs Purchase

table=pd.crosstab(data.marital,data.y)

table.div(table.sum(1).astype(float), axis=0).plot(kind='bar', s

tacked=True)

plt.title('Stacked Bar Chart of Marital Status vs Purchase')

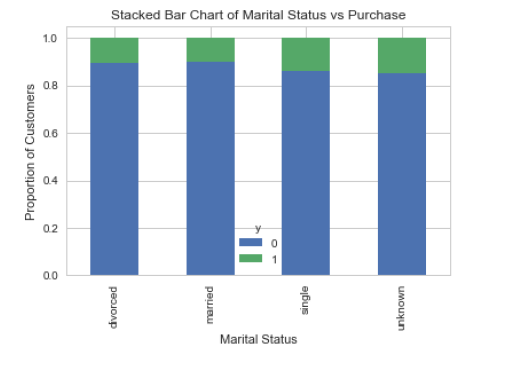
plt.xlabel('Marital Status')

plt.ylabel('Proportion of Customers')

plt.savefig('mariral\_vs\_pur\_stack')

# Hard to see, but the marital status does not seem a strong pre

dictor for the outcome variable.



In [44]: # So, plot Education vs Purchase and check the result

table=pd.crosstab(data.education,data.y)

table.div(table.sum(1).astype(float), axis=0).plot(kind='bar', s

tacked=True)

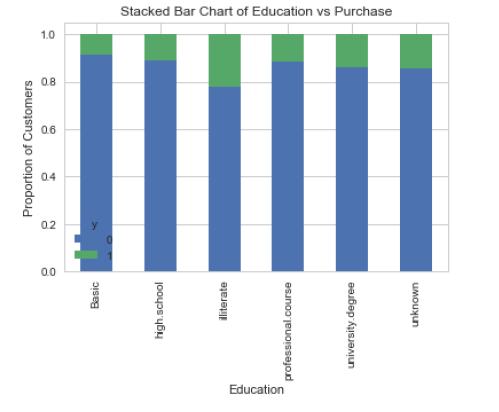
plt.title('Stacked Bar Chart of Education vs Purchase')

plt.xlabel('Education')

plt.ylabel('Proportion of Customers')

plt.savefig('edu\_vs\_pur\_stack')

# Education seems a good predictor of the outcome variable.



In [45]: # Plot the purchase frequency for Day of a week

pd.crosstab(data.day\_of\_week,data.y).plot(kind='bar')

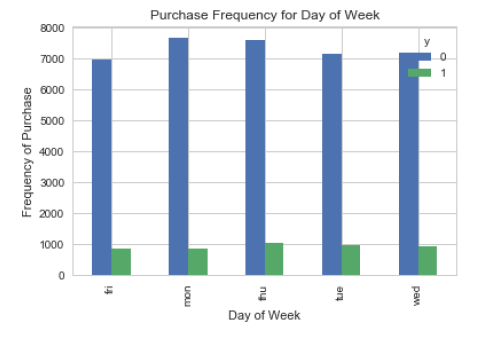
plt.title('Purchase Frequency for Day of Week')

plt.xlabel('Day of Week')

plt.ylabel('Frequency of Purchase')

plt.savefig('pur\_dayofweek\_bar')

# It is not a good predictor also



In [46]: # Ploting prediction for Frequency in a month

pd.crosstab(data.month,data.y).plot(kind='bar')

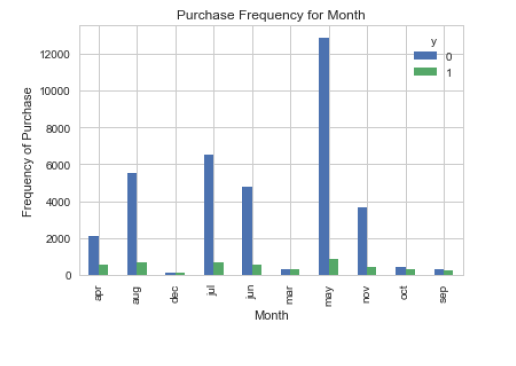
plt.title('Purchase Frequency for Month')

plt.xlabel('Month')

plt.ylabel('Frequency of Purchase')

plt.savefig('pur\_fre\_month\_bar')

# Month might be a good predictor of the outcome variable



In [47]: # Ploting Histogram for age of the customer vs frequency

data.age.hist()

plt.title('Histogram of Age')

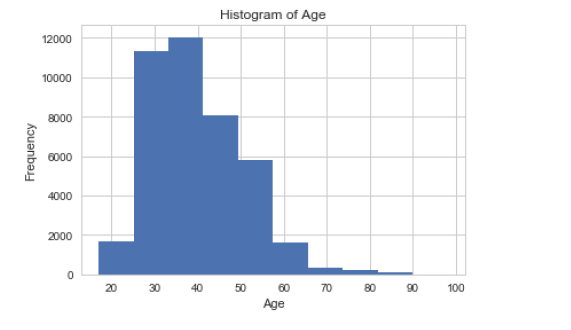
plt.xlabel('Age')

plt.ylabel('Frequency')

plt.savefig('hist\_age')

# Conclusion of graph: The most of the customers of the bank in

this dataset are in the age range of 30-40.



In [97]: # Create dummy variables

cat\_vars=['job','marital','education','default','housing','loan'

,'contact','month','day\_of\_week','poutcome']

data\_vars=data.columns.values.tolist()

to\_keep=[i for i in data\_vars if i not in cat\_vars]

data\_final=data[to\_keep]

data\_final.columns.values

Out[97]: array(['age', 'duration', 'campaign', 'pdays', 'previous', 'e

mp\_var\_rate',

'cons\_price\_idx', 'cons\_conf\_idx', 'euribor3m', 'nr\_em

ployed', 'y'],

dtype=object)

In [112]: cat\_vars=['job','marital','education','default','housing','loan'

,'contact','month','day\_of\_week','poutcome']

data\_vars=data.columns.values.tolist()

to\_keep=[i for i in data\_vars if i not in cat\_vars]

data\_final=data[to\_keep]

data\_final.columns.values

data\_final\_vars=data\_final.columns.values.tolist()

y=data['y']

X=[i for i in data\_final\_vars if i not in y]

#print(data\_final)

# SELF

# print(X)

# print(y)

['age', 'duration', 'campaign', 'pdays', 'previous', 'emp\_var

\_rate', 'cons\_price\_idx', 'cons\_conf\_idx', 'euribor3m', 'nr\_e

mployed', 'y']

0 0

1 0

2 1

3 0

4 1

5 0

6 0

7 0

8 1

9 0

10 0

11 0

12 1

13 0

14 0

15 0

16 1

17 1

18 0

19 0

20 0

21 0

22 0

23 0

24 0

25 0

26 0

27 0

28 0

29 0

..

41158 0

41159 0

41160 0

41161 0

41162 0

41163 0

41164 0

41165 0

41166 0

41167 0

41168 0

41169 0

41170 0

41171 0

41172 1

41173 0

41174 0

41175 0

41176 0

41177 0

41178 1

41179 0

41180 0

41181 0

41182 0

41183 0

41184 0

41185 0

41186 0

41187 0

Name: y, Length: 41188, dtype: int64

In [121]: # SELF

#X=data['age'].tolist()

#X=np.array(X).reshape(-1,1)

#print(X)

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_s

ize=0.3, random\_state=0)

from sklearn.linear\_model import LogisticRegression

from sklearn import metrics

logreg = LogisticRegression()

logreg.fit(X\_train, y\_train)

[[44]

[53]

[28]

...

[42]

[48]

[25]]

Out[121]: LogisticRegression(C=1.0, class\_weight=None, dual=False, fit\_

intercept=True,

intercept\_scaling=1, max\_iter=100, multi\_class='ov

r', n\_jobs=1,

penalty='l2', random\_state=None, solver='liblinea

r', tol=0.0001,

verbose=0, warm\_start=False)

In [124]: #Predicting y-value

y\_pred = logreg.predict(X\_test)

In [125]: #Printing Accuracy

print('Accuracy of logistic regression classifier on test set:

{:.2f}'.format(logreg.score(X\_test, y\_test)))

In [126]: #Cross Validation

from sklearn import model\_selection

from sklearn.model\_selection import cross\_val\_score

kfold = model\_selection.KFold(n\_splits=10, random\_state=7)

modelCV = LogisticRegression()

scoring = 'accuracy'

results = model\_selection.cross\_val\_score(modelCV, X\_train, y\_tr

ain, cv=kfold, scoring=scoring)

print("10-fold cross validation average accuracy: %.3f" % (resul

ts.mean()))

Accuracy of logistic regression classifier on test set: 0.89

10-fold cross validation average accuracy: 0.887

In [135]: from sklearn.metrics import roc\_auc\_score

from sklearn.metrics import roc\_curve

logit\_roc\_auc = roc\_auc\_score(y\_test, logreg.predict(X\_test))

fpr, tpr, thresholds = roc\_curve(y\_test, logreg.predict\_proba(X\_

test)[:,1])

plt.figure()

plt.plot(fpr, tpr, label='Logistic Regression (area = %0.2f)' %

logit\_roc\_auc)

plt.plot([0, 1], [0, 1],'r--')

plt.xlim([0.0, 1.0])

plt.ylim([0.0, 1.05])

plt.xlabel('False Positive Rate')

plt.ylabel('True Positive Rate')

plt.title('Age vs Purchase plot(LOgistic Regression)')

plt.legend(loc="lower right")

plt.savefig('C:/Users/Subhadeep Chakrabort/Desktop/LogisticRegre

ssion-master/LogRegOutput')

plt.show()

