## What is Logistic Regression ?

Logistic Regression is a classification algorithm. It is used to predict a binary outcome (1 / 0, Yes / No, True / False) given a set of independent variables. To represent binary / categorical outcome, we use dummy variables. You can also think of logistic regression as a special case of linear regression when the outcome variable is categorical, where we are using log of odds as dependent variable. In simple words, it predicts the probability of occurrence of an event by fitting data to a logit function.

## Derivation of Logistic Regression Equation

Logistic Regression is part of a larger class of algorithms known as Generalized Linear Model (glm). In 1972, Nelder and Wedderburn proposed this model with an effort to provide a means of using linear regression to the problems which were not directly suited for application of linear regression. Infact, they proposed a class of different models (linear regression, ANOVA, Poisson Regression etc) which included logistic regression as a special case.

The fundamental equation of generalized linear model is:

g(E(y)) = α + βx1 + γx2

Here, g() is the link function, E(y) is the expectation of target variable and α + βx1 + γx2 is the linear predictor ( α,β,γ to be predicted). The role of link function is to ‘link’ the expectation of y to linear predictor.

Important Points

1. GLM does not assume a linear relationship between dependent and independent variables. However, it assumes a linear relationship between link function and independent variables in logit model.
2. The dependent variable need not to be normally distributed.
3. It does not uses OLS (Ordinary Least Square) for parameter estimation. Instead, it uses maximum likelihood estimation (MLE).
4. Errors need to be independent but not normally distributed.

Let’s understand it further using an example:

We are provided a sample of 1000 customers. We need to predict the probability whether a customer will buy (y) a particular magazine or not. As you can see, we’ve a categorical outcome variable, we’ll use logistic regression.

To start with logistic regression, I’ll first write the simple linear regression equation with dependent variable enclosed in a link function:

       g(y) = βo + β(Age)         ---- (a)

Note: For ease of understanding, I’ve considered ‘Age’ as independent variable.

In logistic regression, we are only concerned about the probability of outcome dependent variable ( success or failure). As described above, g() is the link function. This function is established using two things: Probability of Success(p) and Probability of Failure(1-p). p should meet following criteria:

1. It must always be positive (since p >= 0)
2. It must always be less than equals to 1 (since p <= 1)

Now, we’ll simply satisfy these 2 conditions and get to the core of logistic regression. To establish link function, we’ll denote g() with ‘p’ initially and eventually end up deriving this function.

Since probability must always be positive, we’ll put the linear equation in exponential form. For any value of slope and dependent variable, exponent of this equation will never be negative.

p = exp(βo + β(Age)) = e^(βo + β(Age))    ------- (b)

To make the probability less than 1, we must divide p by a number greater than p. This can simply be done by:

p  =  exp(βo + β(Age)) / exp(βo + β(Age)) + 1   =   e^(βo + β(Age)) / e^(βo + β(Age)) + 1    ----- (c)

Using (a), (b) and (c), we can redefine the probability as:

              p = e^y/ 1 + e^y           --- (d)

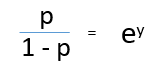
where p is the probability of success. This (d) is the Logit Function

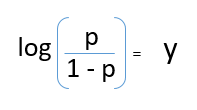
If p is the probability of success, 1-p will be the probability of failure which can be written as:

q = 1 - p = 1 - (e^y/ 1 + e^y)    --- (e)

where q is the probability of failure

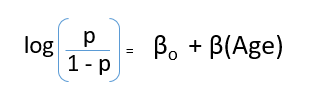
On dividing, (d) / (e), we get,

[](https://www.analyticsvidhya.com/wp-content/uploads/2015/11/1.png)

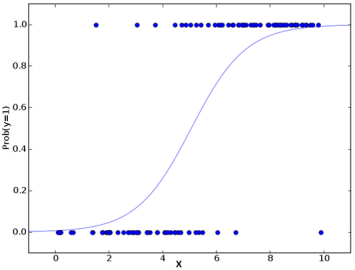
After taking log on both side, we get,  
[](https://www.analyticsvidhya.com/wp-content/uploads/2015/11/2.png)

log(p/1-p) is the link function. Logarithmic transformation on the outcome variable allows us to model a non-linear association in a linear way.

After substituting value of y, we’ll get:

[](https://www.analyticsvidhya.com/wp-content/uploads/2015/11/3.png)

This is the equation used in Logistic Regression. Here (p/1-p) is the odd ratio. Whenever the log of odd ratio is found to be positive, the probability of success is always more than 50%. A typical logistic model plot is shown below. You can see probability never goes below 0 and above 1.

[](https://www.analyticsvidhya.com/wp-content/uploads/2015/11/plot.png)

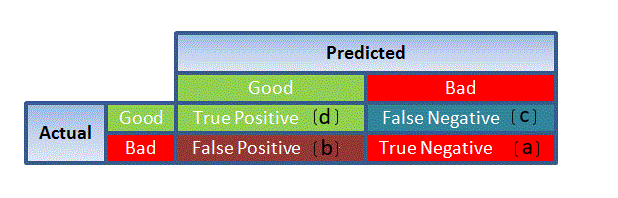
## Performance of Logistic Regression Model

To evaluate the performance of a logistic regression model, we must consider few metrics. Irrespective of tool (SAS, R, Python) you would work on, always look for:

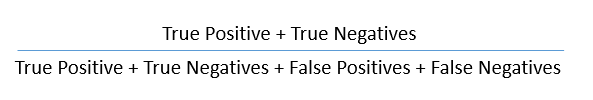
1. **AIC (Akaike Information Criteria)** – The analogous metric of adjusted R² in logistic regression is AIC. AIC is the measure of fit which penalizes model for the number of model coefficients. Therefore, we always prefer model with minimum AIC value.

2. **Null Deviance and Residual Deviance** – Null Deviance indicates the response predicted by a model with nothing but an intercept. Lower the value, better the model. Residual deviance indicates the response predicted by a model on adding independent variables. Lower the value, better the model.

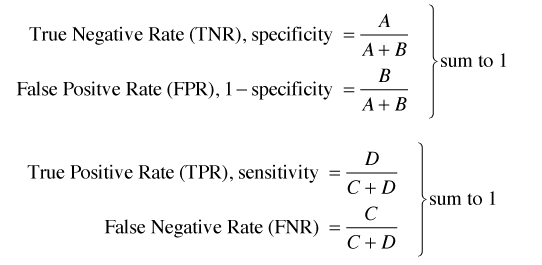
3. **Confusion Matrix:** It is nothing but a tabular representation of Actual vs Predicted values. This helps us to find the accuracy of the model and avoid overfitting. This is how it looks like:

[](https://www.analyticsvidhya.com/wp-content/uploads/2015/11/1111.png)                                                                                            Source: (plug – n – score)

You can calculate the **accuracy** of your model with:

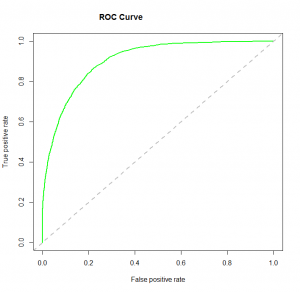
[](https://www.analyticsvidhya.com/wp-content/uploads/2015/11/7.png)

From confusion matrix, Specificity and Sensitivity can be derived as illustrated below:

[](https://www.analyticsvidhya.com/wp-content/uploads/2015/11/9.png)

Specificity and Sensitivity plays a crucial role in deriving ROC curve.

4. **ROC Curve:** Receiver Operating Characteristic(ROC) summarizes the model’s performance by evaluating the trade offs between true positive rate (sensitivity) and false positive rate(1- specificity). For plotting ROC, it is advisable to assume p > 0.5 since we are more concerned about success rate. ROC summarizes the predictive power for all possible values of p > 0.5.  The area under curve (AUC), referred to as index of accuracy(A) or concordance index, is a perfect performance metric for ROC curve. Higher the area under curve, better the prediction power of the model. Below is a sample ROC curve. The ROC of a perfect predictive model has TP equals 1 and FP equals 0. This curve will touch the top left corner of the graph.

[](https://www.analyticsvidhya.com/wp-content/uploads/2015/11/logit_roc.png)

**Note:** For model performance, you can also consider likelihood function. It is called so, because it selects the coefficient values which maximizes the likelihood of explaining the observed data. It indicates goodness of fit as its value approaches one, and a poor fit of the data as its value approaches zero.

## Logistic Regression Model in R

Considering the availability, I’ve build this model on our practice problem – Dressify data set. You can download it [here](http://datahack.analyticsvidhya.com/contest/practice-problem-1). Without going deep into feature engineering, here’s the script of simple logistic regression model:

setwd('C:/Users/manish/Desktop/dressdata')

#load data

train <- read.csv('Train\_Old.csv')

#create training and validation data from given data

install.packages('caTools')

library(caTools)

set.seed(88)

split <- sample.split(train$Recommended, SplitRatio = 0.75)

#get training and test data

dresstrain <- subset(train, split == TRUE)

dresstest <- subset(train, split == FALSE)

#logistic regression model

model <- glm (Recommended ~ .-ID, data = dresstrain, family = binomial)

summary(model)

predict <- predict(model, type = 'response')

#confusion matrix

table(dresstrain$Recommended, predict > 0.5)

#ROCR Curve

library(ROCR)

ROCRpred <- prediction(predict, dresstrain$Recommended)

ROCRperf <- performance(ROCRpred, 'tpr','fpr')

plot(ROCRperf, colorize = TRUE, text.adj = c(-0.2,1.7))

#plot glm

library(ggplot2)

ggplot(dresstrain, aes(x=Rating, y=Recommended)) + geom\_point() +

stat\_smooth(method="glm", family="binomial", se=FALSE)

This data require lots of cleaning and feature engineering. The scope of this article restricted me to keep the example focused on building logistic regression model. This data is [available](http://datahack.analyticsvidhya.com/contest/practice-problem-1) for practice. I’d recommend you to work on this problem. There’s a lot to learn.

**Introduction**

Logistic Regression is a Machine Learning classification algorithm that is used to predict the probability of a categorical dependent variable. In logistic regression, the dependent variable is a binary variable that contains data coded as 1 (yes, success, etc.) or 0 (no, failure, etc.). In other words, the logistic regression model predicts P(Y=1) as a function of X.

**Logistic Regression 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.

Keeping the above assumptions in mind, let’s look at our dataset.

**Data**

The dataset comes from the [UCI Machine Learning repository](http://archive.ics.uci.edu/ml/index.php), and 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 can be downloaded from [here](https://raw.githubusercontent.com/madmashup/targeted-marketing-predictive-engine/master/banking.csv).

**Methods**

|  |  |
| --- | --- |
| [**decision\_function**](http://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html#sklearn.linear_model.LogisticRegression.decision_function)(X) | Predict confidence scores for samples. |
| [**densify**](http://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html#sklearn.linear_model.LogisticRegression.densify)() | Convert coefficient matrix to dense array format. |
| [**fit**](http://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html#sklearn.linear_model.LogisticRegression.fit)(X, y[, sample\_weight]) | Fit the model according to the given training data. |
| [**get\_params**](http://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html#sklearn.linear_model.LogisticRegression.get_params)([deep]) | Get parameters for this estimator. |
| [**predict**](http://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html#sklearn.linear_model.LogisticRegression.predict)(X) | Predict class labels for samples in X. |
| [**predict\_log\_proba**](http://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html#sklearn.linear_model.LogisticRegression.predict_log_proba)(X) | Log of probability estimates. |
| [**predict\_proba**](http://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html#sklearn.linear_model.LogisticRegression.predict_proba)(X) | Probability estimates. |
| [**score**](http://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html#sklearn.linear_model.LogisticRegression.score)(X, y[, sample\_weight]) | Returns the mean accuracy on the given test data and labels. |
| [**set\_params**](http://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html#sklearn.linear_model.LogisticRegression.set_params)(\*\*params) | Set the parameters of this estimator. |
| [**sparsify**](http://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html#sklearn.linear_model.LogisticRegression.sparsify)() | Convert coefficient matrix to sparse format. |

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)

The dataset provides the bank customers’ information. It includes 41,188 records and 21 fields.

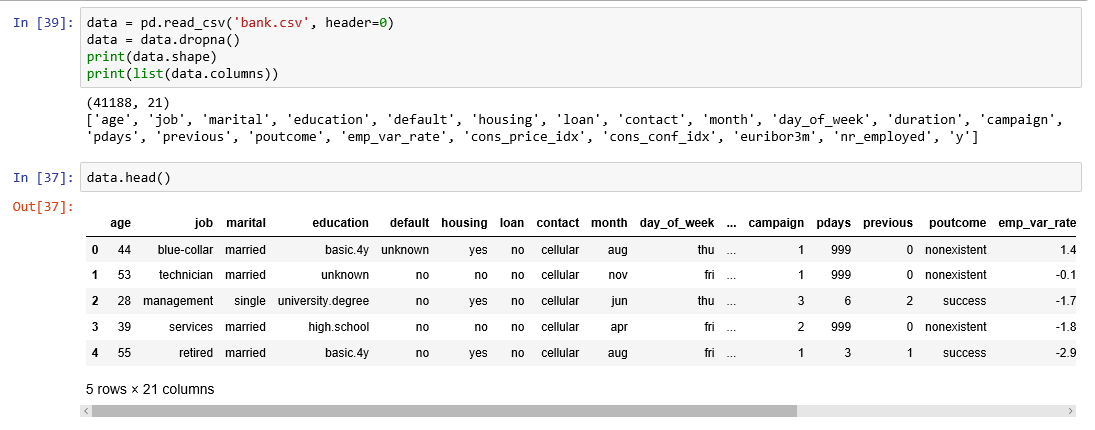


Figure 1

**Input variables**

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”)
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”)
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’). 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”)
16. emp.var.rate: employment variation rate — (numeric)
17. cons.price.idx: consumer price index — (numeric)
18. cons.conf.idx: consumer confidence index — (numeric)
19. euribor3m: euribor 3 month rate — (numeric)
20. nr.employed: number of employees — (numeric)

**Predict variable (desired target):**

y — has the client subscribed a term deposit? (binary: “1”, means “Yes”, “0” means “No”)

The education column of the dataset has many categories and we need to reduce the categories for a better modelling. The education column has the following categories:

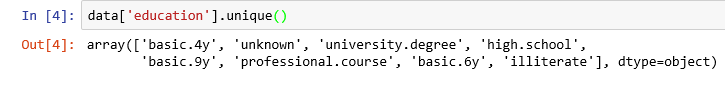


Figure 2

Let us group “basic.4y”, “basic.9y” and “basic.6y” together and call them “basic”.

data['education']=np.where(data['education'] =='basic.9y', 'Basic', data['education'])  
data['education']=np.where(data['education'] =='basic.6y', 'Basic', data['education'])  
data['education']=np.where(data['education'] =='basic.4y', 'Basic', data['education'])

After grouping, this is the columns:

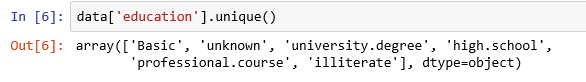


Figure 3

**Data exploration**

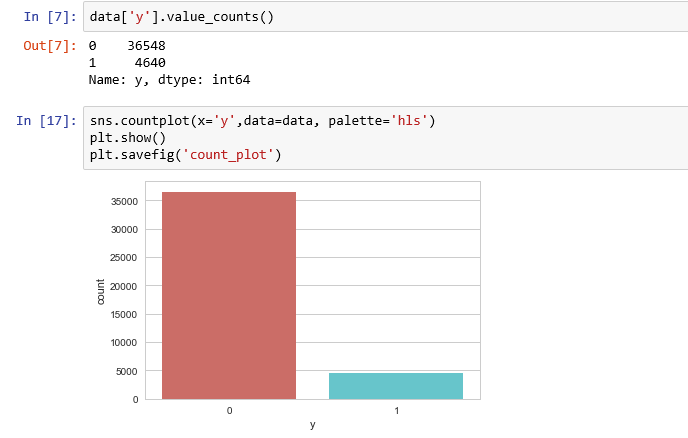


Figure 4

There are 36548 no’s and 4640 yes’s in the outcome variables.

Let’s get a sense of the numbers across the two classes.

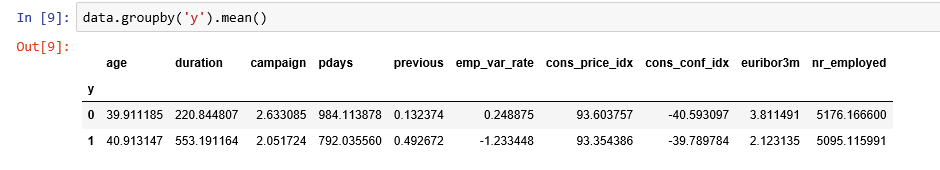


Figure 5

***Observations***:

* The average age of customers who bought the term deposit is higher than that of the customers who didn’t.
* The pdays (days since the customer was last contacted) is understandably lower for the customers who bought it. The lower the pdays, the better the memory of the last call and hence the better chances of a sale.
* Surprisingly, campaigns (number of contacts or calls made during the current campaign) are lower for customers who bought the term deposit.

We can calculate categorical means for other categorical variables such as education and marital status to get a more detailed sense of our data.

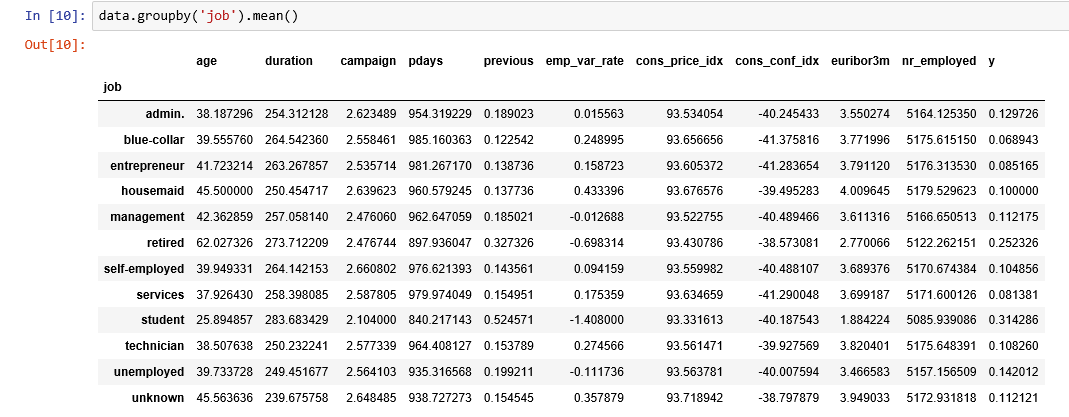


Figure 6

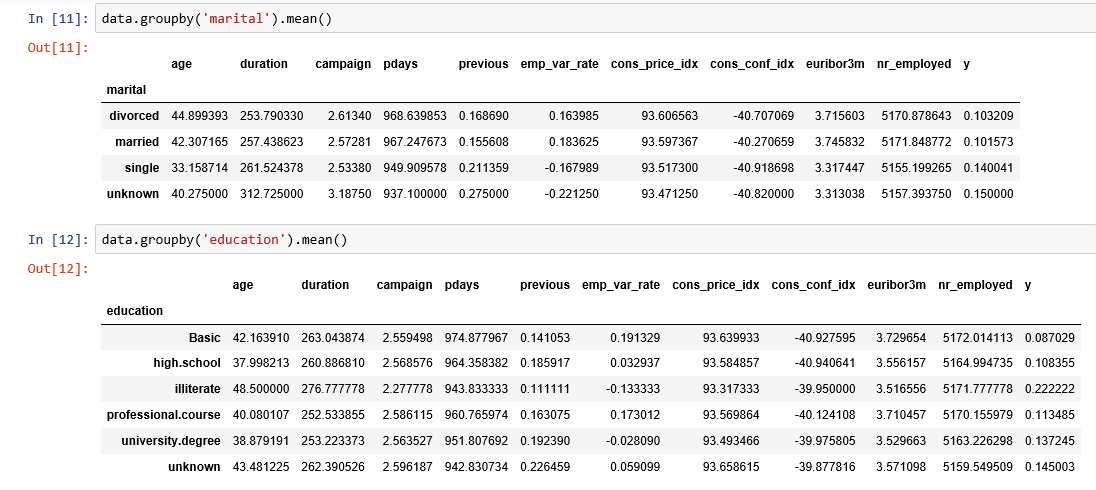


Figure 7

**Visualizations**

%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')

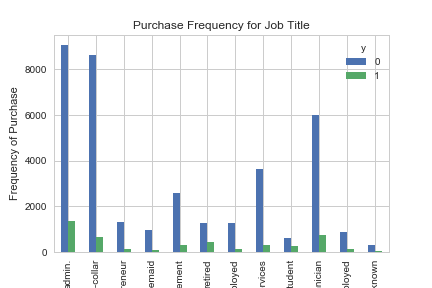


Figure 8

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 variable.

table=pd.crosstab(data.marital,data.y)  
table.div(table.sum(1).astype(float), axis=0).plot(kind='bar', stacked=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')

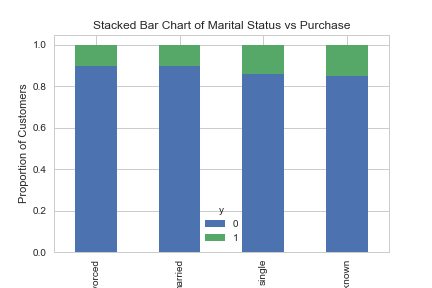


Figure 9

The marital status does not seem a strong predictor for the outcome variable.

table=pd.crosstab(data.education,data.y)  
table.div(table.sum(1).astype(float), axis=0).plot(kind='bar', stacked=True)  
plt.title('Stacked Bar Chart of Education vs Purchase')  
plt.xlabel('Education')  
plt.ylabel('Proportion of Customers')  
plt.savefig('edu\_vs\_pur\_stack')

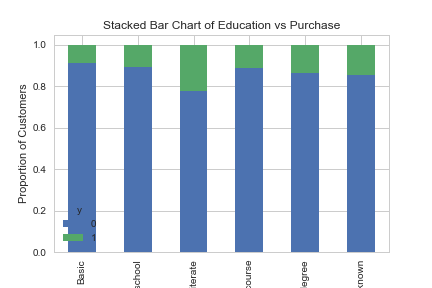


Figure 10

Education seems a good predictor of the outcome variable.

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')

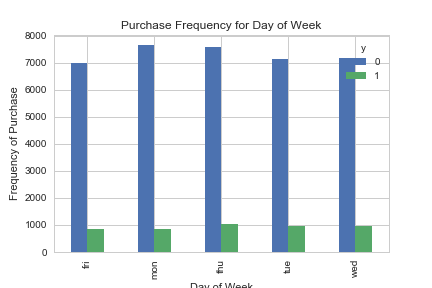


Figure 11

Day of week may not be a good predictor of the outcome.

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')

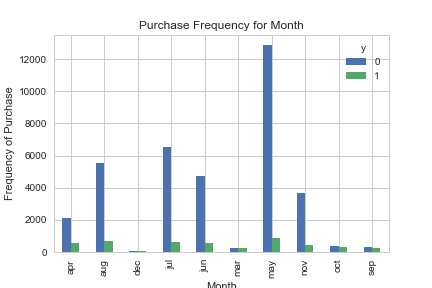


Figure 12

Month might be a good predictor of the outcome variable.

data.age.hist()  
plt.title('Histogram of Age')  
plt.xlabel('Age')  
plt.ylabel('Frequency')  
plt.savefig('hist\_age')

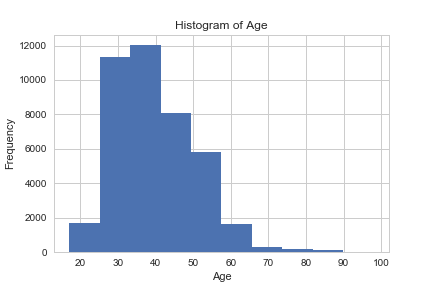


Figure 13

Most of the customers of the bank in this dataset are in the age range of 30–40.

pd.crosstab(data.poutcome,data.y).plot(kind='bar')  
plt.title('Purchase Frequency for Poutcome')  
plt.xlabel('Poutcome')  
plt.ylabel('Frequency of Purchase')  
plt.savefig('pur\_fre\_pout\_bar')

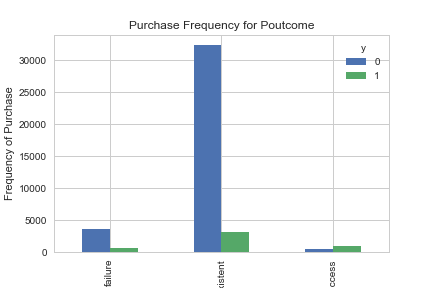


Figure 14

Poutcome seems to be a good predictor of the outcome variable.

**Create dummy variables**

That is variables with only two values, zero and one.

cat\_vars=['job','marital','education','default','housing','loan','contact','month','day\_of\_week','poutcome']  
for var in cat\_vars:  
 cat\_list='var'+'\_'+var  
 cat\_list = pd.get\_dummies(data[var], prefix=var)  
 data1=data.join(cat\_list)  
 data=data1

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]

Our final data columns will be:

data\_final=data[to\_keep]  
data\_final.columns.values

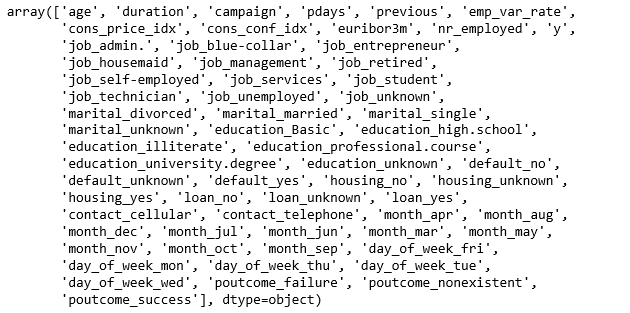


Figure 15

data\_final\_vars=data\_final.columns.values.tolist()  
y=['y']  
X=[i for i in data\_final\_vars if i not in y]

**Feature Selection**

[Recursive Feature Elimination (RFE)](http://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.RFE.html) is based on the idea to repeatedly construct a model and choose either the best or worst performing feature, setting the feature aside and then repeating the process with the rest of the features. This process is applied until all features in the dataset are exhausted. The goal of RFE is to select features by recursively considering smaller and smaller sets of features.

from sklearn import datasets  
from sklearn.feature\_selection import RFE  
from sklearn.linear\_model import LogisticRegression

logreg = LogisticRegression()

rfe = RFE(logreg, 18)  
rfe = rfe.fit(data\_final[X], data\_final[y] )  
print(rfe.support\_)  
print(rfe.ranking\_)

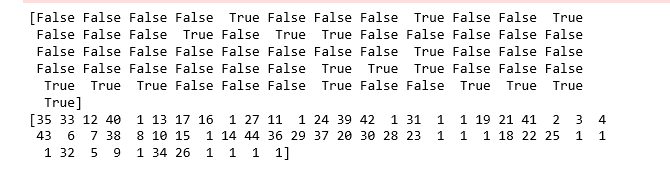


Figure 16

The RFE has helped us select the following features: “previous”, “euribor3m”, “job\_blue-collar”, “job\_retired”, “job\_services”, “job\_student”, “default\_no”, “month\_aug”, “month\_dec”, “month\_jul”, “month\_nov”, “month\_oct”, “month\_sep”, “day\_of\_week\_fri”, “day\_of\_week\_wed”, “poutcome\_failure”, “poutcome\_nonexistent”, “poutcome\_success”.

cols=["previous", "euribor3m", "job\_blue-collar", "job\_retired", "job\_services", "job\_student", "default\_no",   
 "month\_aug", "month\_dec", "month\_jul", "month\_nov", "month\_oct", "month\_sep", "day\_of\_week\_fri", "day\_of\_week\_wed",   
 "poutcome\_failure", "poutcome\_nonexistent", "poutcome\_success"]   
X=data\_final[cols]  
y=data\_final['y']

**Implementing the model**

import statsmodels.api as sm  
logit\_model=sm.Logit(y,X)  
result=logit\_model.fit()  
print(result.summary())

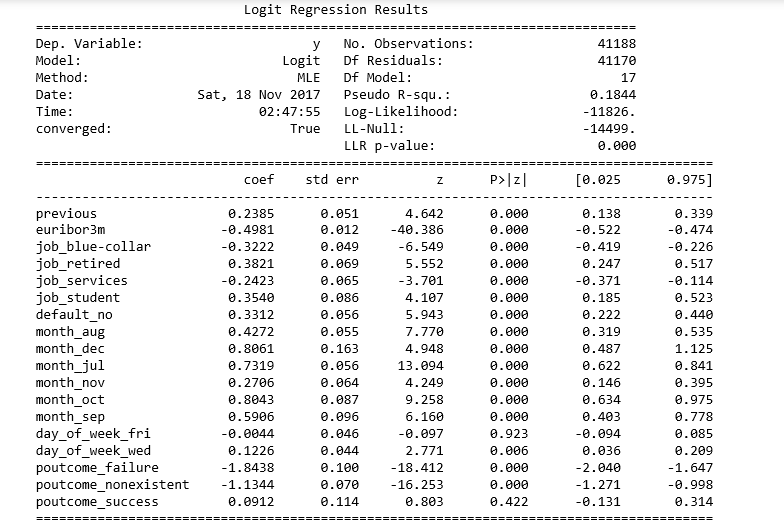


Figure 17

The p-values for most of the variables are smaller than 0.05, therefore, most of them are significant to the model.

**Logistic Regression Model Fitting**

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.3, random\_state=0)  
from sklearn.linear\_model import LogisticRegression  
from sklearn import metrics  
logreg = LogisticRegression()  
logreg.fit(X\_train, y\_train)

***LogisticRegression(C=1.0, class\_weight=None, dual=False, fit\_intercept=True, intercept\_scaling=1, max\_iter=100, multi\_class=’ovr’, n\_jobs=1, penalty=’l2', random\_state=None, solver=’liblinear’, tol=0.0001,  
verbose=0, warm\_start=False***

**Predicting the test set results and calculating the accuracy**

y\_pred = logreg.predict(X\_test)  
print('Accuracy of logistic regression classifier on test set: {:.2f}'.format(logreg.score(X\_test, y\_test)))

***Accuracy of logistic regression classifier on test set: 0.90***

**Cross Validation**

Cross validation attempts to avoid overfitting while still producing a prediction for each observation dataset. We are using 10-fold Cross-Validation to train our Logistic Regression model.

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\_train, cv=kfold, scoring=scoring)  
print("10-fold cross validation average accuracy: %.3f" % (results.mean()))

***10-fold cross validation average accuracy: 0.897***

The average accuracy remains very close to the Logistic Regression model accuracy; hence, we can conclude that our model generalizes well.

**Confusion Matrix**

from sklearn.metrics import confusion\_matrix  
confusion\_matrix = confusion\_matrix(y\_test, y\_pred)  
print(confusion\_matrix)

***[[10872 109]  
 [ 1122 254]]***

The result is telling us that we have 10872+254 correct predictions and 1122+109 incorrect predictions.

**Compute precision, recall, F-measure and support**

To quote from [Scikit Learn](http://scikit-learn.org/stable/index.html" \t "_blank):

The precision is the ratio tp / (tp + fp) where tp is the number of true positives and fp the number of false positives. The precision is intuitively the ability of the classifier to not label a sample as positive if it is negative.

The recall is the ratio tp / (tp + fn) where tp is the number of true positives and fn the number of false negatives. The recall is intuitively the ability of the classifier to find all the positive samples.

The F-beta score can be interpreted as a weighted harmonic mean of the precision and recall, where an F-beta score reaches its best value at 1 and worst score at 0.

The F-beta score weights the recall more than the precision by a factor of beta. beta = 1.0 means recall and precision are equally important.

The support is the number of occurrences of each class in y\_test.

from sklearn.metrics import classification\_report  
print(classification\_report(y\_test, y\_pred))

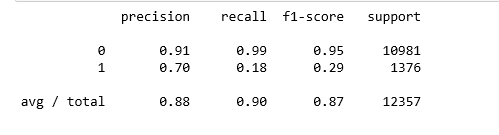


Figure 18

**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.

**ROC Curve**

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('Receiver operating characteristic')  
plt.legend(loc="lower right")  
plt.savefig('Log\_ROC')  
plt.show()

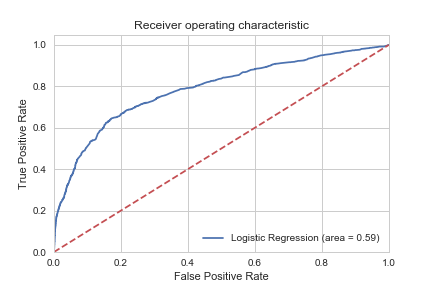


Figure 19

[The receiver operating characteristic (ROC)](https://en.wikipedia.org/wiki/Receiver_operating_characteristic) curve is another common tool used with binary classifiers. The dotted line represents the ROC curve of a purely random classifier; a good classifier stays as far away from that line as possible (toward the top-left corner).