Machine Learning and Data Mining
Lab 4: Logistic Regression

Instructions

(1) In this exercise you will use `jupyter-notebook` or `google collaboratory` and `python>=3.5`. Type your code, display the outputs, and write your answers to the questions asked in the notebook.
(2) Upload PDF of your notebook file via *Juno*. Typeset your name, roll number and section on the top of the file. Also, filename has to specified as "your_first_name_last_name_roll_number.PDF".)

Learning Outcomes

- Perform classification using Logistic Regression
- Identify statistical significance of variables using p-values
- Evaluate model accuracy using *Confusion Matrix*

Stock Market Data

You will examine and create models of the `Weekly` data set which contains data of stock market. This data consists of percentage returns for the S&P 500 stock index over a period of 1089 weeks (that is, 21 years) from the beginning of 1990 to the end of 2010.

For each date, we have recorded the percentage returns for each of the five previous trading days, `Lag1` through `Lag5`. We have also recorded `Volume` (the number of shares traded on the previous day, in billions), `Today` (the percentage return on the date in question) and `Direction` (whether the market was Up or Down on this date).

(You can think `Lag1` as Monday, `Lag2` as Tuesday, `Lag3` as Wednesday, `Lag4` as Thursday and `Lag5` as Friday; row 1 contains return from week-1 in percentage, row-2 contains return from week-2, and so on.)

You will perform logistic regression to solve the binary classification task. Logistic function is given as: $p(X) = \frac{e^{\beta 0+\beta 1X}}{1+ e^{\beta 0+\beta 1X}}$. Here $\{\beta\}$ are the coefficients to be determined. The decision boundary separating the two classes in this model can be determined by choosing *p(X) = 0.5.* The equation of the decision boundary is given by: $\beta_0 + \beta_1 X = 0$, which separates the positive and negative class examples.

Answer the following questions.

(1) (a) Produce some numerical summaries of the `Weekly` data. Are there any null values in the data?

(b) Produce some graphical summaries of the `Weekly` data. Explore the following. Does there appear to be any patterns? Is there correlation between Today's return and previous day's return? Comment on the `Volume-Year` relation.

(2) (a) What is the datatype of the class variable `Direction`?

(b) Since we wish to perform logistic regression on `Direction`, we have to first convert it into a numeric datatype. So, we will create a numeric predictor called `nDirection` such that if `Direction` is `Down` then `nDirection=0`, and if `Direction` is `Up` then `nDirection=1`. Note that this variable is often referred as a *dummy* variable in textbooks.

Create a user defined function `numeric_direction(argument)` that performs the above conversion task. Then invoke this function as follows to create the new predictor:

```
df['nDirection']=df['Direction'].apply(numeric_direction)
```

Verify the dataframe using `df.info()` and `df.head()`.

(c) Compute the correlation coefficients between a pair of data columns and display as a matrix using `plt.matshow(df.corr().abs())`.

(3) Use the full data set to perform a logistic regression with `nDirection` as the response and the five lag variables plus `Volume` as predictors. Name the model as `model_1`.

Use the summary function to print the results.

Do any of the predictors appear to be statistically significant? If so, which ones?

(4) Compute the confusion matrix and overall fraction of correct predictions. Interpret the confusion matrix by explaining the types of mistakes made by logistic regression. (Make use of the following tables.)

Table-1: Possible results from a classifier.

		Predicted class		
		− or Null	+ or Non-null	Total
True class	− or Null	True Neg. (TN)	False Pos. (FP)	N
	+ or Non-null	False Neg. (FN)	True Pos. (TP)	P
	Total	N*	P*	

Table-2: Measures of Classification

Name	Definition	Synonyms
False Pos. rate	FP/N	Type I error, 1−Specificity
True Pos. rate	TP/P	1−Type II error, power, sensitivity, recall
Pos. Pred. value	TP/P*	Precision, 1−false discovery proportion
Neg. Pred. value	TN/N*	

(5) Create another model by dropping the irrelevant predictors. Plot the data of `Lag2` vs. `nDirection`, and overlay model predictions on the same plot.

What is the equation of the decision boundary?

Explain your view on the applicability of logistic regression on this data set.

SOLUTION (1)

In [ ]:

```
import pandas as pd
```

In [ ]:

```
df = pd.read_csv(_____________) # Read CSV file Weekly.csv to a dataframe object.
Fill the blank.
print(df.head())
print(df.info())
print(df.describe())

##Check if there are null values in the data, and if so, how many rows have null
values.
df[df.isnull()==True].count()
```

In [ ]:

```
import matplotlib.pyplot as plt
%matplotlib inline
#Plot scatter matrix plot of the dataframe
pd.plotting.scatter_matrix(________, figsize=________)
#Compute the correlation coefficients of a pair of predictors
df.corr()
```

Do you see any patterns? Which feature-pairs show pattern?

SOLUTION (2)

In [ ]:

```
# Define a function to convert the 'Up' and 'Down' values to 1 and 0 respectivel
y. Fill the blank spaces.
____ numeric_direction (value):
    if value == 'Up':
        return _______
    if value == 'Down':
        return _______

#Test the function if it is generating correct values
print ("Value of Up is %d" % numeric_direction('Up'))
print ("Value of Down is %d" % numeric_direction('Down'))
```

In [ ]:

```
df['nDirection'] = df['Direction'].apply(numeric_direction)
df.head()
```

In [ ]:

```
#Delete 'Direction' column from the dataframe.
#In the drop function below, 0 stands for a row, and 1 stands for column.
#Since we will delete a column, we specify 1 in the second argument
df = df.drop('Direction', 1)
df.head()
```

In [ ]:

```
#Previously the scatter matrix plot didnot display 'Direction' as it has a non-n
umeric datatype.
wcorr = df.corr()
plt.matshow(wcorr.abs())
plt.colorbar()
plt.xticks(range(len(wcorr.columns)), wcorr.columns, rotation='vertical');
plt.yticks(range(len(wcorr.columns)), wcorr.columns);

#Another way of displaying the correlations
wcorr.abs().style.background_gradient()

#So, now we can see correlations involving'nDirection'
```

SOLUTION (3)

Reference:
https://www.statsmodels.org/devel/generated/statsmodels.discrete.discrete_model.Logit.html#statsmodels.c
(https://www.statsmodels.org/devel/generated/statsmodels.discrete.discrete_model.Logit.html#statsmodels.

In [ ]:

```
import statsmodels.formula.api as sm
#Fit the logistic regression model.
model_1 = sm.logit('nDirection ~ Lag1 + Lag2 + Lag3 + Lag4 + Lag5 + Volume', df)
.fit()
```

In [ ]:

```
print(dir(model_1))
#Print the summary of the results of model_1. Fill the blank.
model_1.____________
```

In [ ]:

```
#p-value tell us about statistical significant association between individual pr
edictors and response
# Find the predictors that are statistically significant. Fill the blank below.
model_1.____________
```

SOLUTION (4)

In [ ]:

```
confusion_matrix = pd.DataFrame(model_1.pred_table())
print (confusion_matrix)
```

In [ ]:

```
#Give names to the columns
confusion_matrix.columns = ['Predicted_Class_0', 'Predicted_Class_1']
confusion_matrix.index = ['True_Class_0', 'True_Class_1']
print (confusion_matrix)
```

In [ ]:

```
sum_tmp = confusion_matrix.sum(axis=1)
print(sum_tmp, sum_tmp.shape)
```

In [ ]:

```
sum_tmp_1 = confusion_matrix.sum(axis=0)
print(sum_tmp_1, sum_tmp_1.shape)
```

In [ ]:

```
confusion_matrix['Total'] = sum_tmp
confusion_matrix.loc['Total'] = sum_tmp_1
print(confusion_matrix)
```

How many mistakes are committed by the model?

SOLUTION (5)

Reference:
[https://www.statsmodels.org/devel/generated/statsmodels.discrete.discrete_model.Logit.html#statsmodels.c
(https://www.statsmodels.org/devel/generated/statsmodels.discrete.discrete_model.Logit.html#statsmodels.c](https://www.statsmodels.org/devel/generated/statsmodels.discrete.discrete_model.Logit.html#statsmodels.c)

In [ ]:

```
#Fit a logistic model to 'nDirection' using only one predictor 'Lag2'. Fill the
blank.
model_2 = _________________
print(model_2.summary())
print("---------------")
#print(dir(model_2))
print("---- Confusion Table ----")
print(pd.DataFrame(model_2.pred_table()))
```

Plot the data and model prediction

In [ ]:

```
import numpy as np

plt.plot(df.Lag2, df.nDirection, 'ro', alpha=0.2, label='data')
plt.xlabel('Lag2')
plt.ylabel('nDirection: Probabililty of Up')

x1 = df.Lag2
y1 = model_2.predict(x1)
#print (y1)
#Note below we have to sort the data because without sorting the data will be ra
ndomly connected by lines.
x2, y2 = zip(*sorted(zip(x1, y1)))
plt.plot(x2, y2, 'b--', label='Fit')
plt.legend(loc='best')
```

Note that in this problem logistic regression does not perform well.