```
#Topic:Logistic using Wine Data Set
#-----------------------------
#libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split

#dataset - understand it first : target class based on other parameters
#https://scikit-learn.org/stable/modules/generated/sklearn.datasets.load_wine.ht
ml
from sklearn.datasets import load_wine
wine = load_wine()
wine
wine.data

# feature matrix
X = wine.data
X
# target vector
y = wine.target
y
# class labels
labels = wine.feature_names
labels

#join X & y to dataframe
df = pd.DataFrame(wine.data)
#name the columns
df.columns = ['Alcohol', 'Malicacid', 'Ash', 'AlcalinityOfAsh', 'Magnesium',
'TotalPhenols', 'Flavanoids', 'NonflavanoidPhenols', 'Proanthocyanins',
'ColorIntensity', 'Hue', 'OD280_OD315', 'Proline']
df
df['target'] = pd.Series(wine.target) #add class column as target
df

#%%% what to do in this assignment
##in this case example Alcohol, Magnesium, Proline are used as IV to predict two
target class, you required to select 3 different sets of variables and then
predict the class
#you will have use different variables, create test data accordingly for
prediction
#%%%
#select class 0 & 1 from target column
df.shape
df.target.value_counts()
data = df[df.target != 2]
data.target.value_counts()
#use only 3 columns for constructing a model
data2 = data[['Alcohol', 'Magnesium', 'Proline', 'target']]
data2.head()
data2.describe()
```

```
#%%plots
#barplot
data2.target.value_counts().plot.bar()
plt.show();

#histogram
data2.Alcohol.plot.hist()
plt.show();

#density
data2.Alcohol.plot.density()
plt.show();

#correlation plot
corr = data2.corr()
sns.heatmap(corr, annot=True)
plt.show();


#%%predict target
data2.sample()
newdata = pd.DataFrame({'Alcohol':[12,13], 'Magnesium':[72,100], 'Proline':[300,
1000]})
newdata

#X & y
data2.head()
X = data2[['Alcohol', 'Magnesium', 'Proline']].values
X
y = data2['target'].values
y
data2.columns
#train and test
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,
random_state=1)
X_train.shape, X_test.shape

#%%% Logistic Regression
#libraries
import matplotlib.pyplot as plt
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report, confusion_matrix
import seaborn as sns

#data
X_train, X_test, y_train, y_test
X_train.shape, X_test.shape

#Logistic model
logModel = LogisticRegression(random_state=0)
logModel.fit(X_train, y_train)
logModel.classes_
logModel.intercept_
logModel.coef_  #odd ratio
```

```
#Evaluate Model
X_test
logModel.predict_proba(X_test)  #prob of 0 & 1 class
logModel.predict(X_test)
y_predLG = logModel.predict(X_test)
logModel.score(X=X_train, y=y_train)
logModel.score(X=X_test, y=y_test)
cm=confusion_matrix(y_true= y_test, y_pred=y_predLG )
#TN FP
#FN TP
#we want more TN(Acutal-0, Predict-0) & TP
#FN (Actual-1, Predict-0)
cm
sns.heatmap(cm, annot=True)
plt.show();
#report
classification_report(y_test, logModel.predict(X_test))

#predict

y_pred2B = logModel.predict(newdata)
y_pred2B
y_pred2C = logModel.predict_proba(newdata)
y_pred2C

#%%using stats model
import statsmodels.api as sm
X_train = sm.add_constant(X)
X_train
#logit = sm.Logit(y=y_train, X=X_train)
X_train.shape
y_train.shape
y_train = y_train.reshape(-1,1)
logit = sm.Logit(endog=y_train, exog=X_train)

logit.fit().params
# fit the model
result = logit.fit()
#Interpreting the results
result.summary()
result.summary2() #what is the value of AIC - 59.81 here
result.conf_int()
result.predict(X_test)
y_predSM = result.predict(X_test)
y_predSM
result.predict_class_
confusion_matrix(y_true= y_test, y_pred=y_predSM )
#with constant
logit2 = sm.Logit(endog=y_train, exog=sm.add_constant(X_train))
logit2.fit().params
# fit the model
result2 = logit2.fit()
#Interpreting the results
```

```
result2.summary()
result2.summary2() #what is the value of AIC - 59.81 here
result2.conf_int()
result2.pred_table(threshold=.5)
X_train.shape
X_test.shape
#predict test data
result2.predict(sm.add_constant(X_test))

#new data
result2.predict(sm.add_constant(newdata)) #.9 ~ 1; #.00 ~ 0
logModel.predict_proba(newdata)
logModel.predict(newdata)

#Logistic Regression is the base of all classification algorithms.
#A good understanding on logistic regression and goodness of fit measures will
really help in understanding complex machine learning algorithms like neural
networks and SVMs.
#One has to be careful while selecting the model, all the goodness of fit
measures are calculated on training data. We may have to do cross validation to
get an idea on the test error.

#Mention 2 applications of Logistic Regression (domain and what can we predict)
#end here
```