**import** **pandas** **as** **pd**

data = pd.read\_csv(‘path’)

**from** **sklearn.datasets** **import**

load\_iris iris = load\_iris()

print(iris.feature\_names)

print(iris.target\_names)

print(iris.target) *#vector format 0,1,2 num*

print(iris.data) *#matrix format*

print(type(iris.data)) *#numpy.ndarray*

Fitting a machine learning model

*#1. import the class*

*#2. instantiate*

*#3. fit the model*

*#KNN - plot (x,y,color)*

*- mark territory for color*

*- territory predict the new point*

*http://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html*

**from** **sklearn.neighbors** **import** KNeighborsClassifier

knn1 = KNeighborsClassifier(n\_neighbors=1)

print(knn1)

KNeighborsClassifier

(algorithm='auto', leaf\_size=30, metric='minkowski', metric\_params=None, n\_jobs=1, n\_neighbors=1, p=2, weights='uniform')

knn1.fit(X,y)

prediction = knn1.predict([[2,4,3,1]])

print(prediction)

type(prediction)

print(iris.target\_names)

prediction =

knn1.predict([[2,4,3,1],[4,6,5,3]])

print(prediction)

knn5 = KNeighborsClassifier(n\_neighbors=5)

knn5.fit(X,y)

prediction5 =

knn5.predict([[2,4,3,1],[4,6,5,3]])

print(prediction5)

*#logistic Regration - classification*

**from** **sklearn.linear\_model** **import** LogisticRegression

logisticreg = LogisticRegression()

logisticreg.fit(X,y)

predict\_lr =

logisticreg.predict([[2,4,3,1],[4,6,5,3]])

print(predict\_lr)

prediction = knn.predict([[5.1,3.5,1.4,0.2],[6.3,3.3,4.7,1.6]])

print(prediction)

prediction5 = knn5.predict([[5.1,3.5,1.4,0.2],[6.3,3.3,4.7,1.6]])

print(prediction5)

prediction\_lr = logisticreg.predict(([[5.1,3.5,1.4,0.2],[6.3,3.3,4.7,1.6]]))

print(prediction\_lr)

Train – Test

**import** **numpy** **as** **np**

**from** **sklearn.model\_selection** **import** train\_test\_split

X\_train, X\_test, y\_train, y\_test =

train\_test\_split(X, y, test\_size=0.3)

print(X.shape)

print(X\_train.shape)

print(X\_test.shape)

print(y\_train.shape)

print(y\_test.shape)

**from** **sklearn.linear\_model** **import** LogisticRegression

logisticreg = LogisticRegression()

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

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

*#accuracy\_score*

**from** **sklearn** **import** metrics

print(metrics.accuracy\_score(y\_test,y\_pred))

**from** **sklearn.neighbors** **import** KNeighborsClassifier

knn1 = KNeighborsClassifier(n\_neighbors=1)

knn1.fit(X\_train,y\_train)

y\_pred = knn1.predict(X\_test)

print(metrics.accuracy\_score(y\_test,y\_pred))

**from** **sklearn.neighbors** **import** KNeighborsClassifier

knn5 = KNeighborsClassifier(n\_neighbors=5)

knn5.fit(X\_train,y\_train)

y\_pred = knn5.predict(X\_test)

print(metrics.accuracy\_score(y\_test,y\_pred))

**from** **sklearn.model\_selection** **import** train\_test\_split

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.4, random\_state=4)

**from** **sklearn.linear\_model** **import** LogisticRegression

logisticreg = LogisticRegression()

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

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

**from** **sklearn** **import** metrics

print(metrics.accuracy\_score(y\_test,y\_pred))

**from** **sklearn.neighbors** **import** KNeighborsClassifier

knn1 = KNeighborsClassifier(n\_neighbors=1)

knn1.fit(X\_train,y\_train)

y\_pred = knn1.predict(X\_test)

print(metrics.accuracy\_score(y\_test,y\_pred))

**from** **sklearn.neighbors** **import** KNeighborsClassifier

knn5 = KNeighborsClassifier(n\_neighbors=5)

knn5.fit(X\_train,y\_train)

y\_pred = knn5.predict(X\_test)

print(metrics.accuracy\_score(y\_test,y\_pred))

knn\_scores=[]

**for** i **in** range(1,31):

knn=KNeighborsClassifier(n\_neighbors=i)

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

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

knn\_scores.append(scores.mean())

print(knn\_scores)

**import** **matplotlib.pyplot** **as** **plt**

%matplotlib inline

plt.plot(range(1,31),knn\_scores)

plt.xlabel('Accuracy\_score')

plt.ylabel('n\_neighbors')

**import** **pandas** **as** **pd**

*#pipe delimited*

reads = pd.read\_table('path/.txt’)

reads.head()

reads = pd.read\_table('path/.txt’, sep='|')

reads.head()

*#tsv*

employee = pd.read\_table('path/.tsv') employee.head()

*#choose certain columns*

cols=['Name','Position']

employee = pd.read\_table('path/.tsv')

employee[cols].head()

*#choose certain rows*

employee =

pd.read\_table('path/.tsv', nrows = 4)

employee

*#display datatypes*

employee = pd.read\_table('path/.tsv')

employee.dtypes

*#display integer datatypes*

**import** **numpy** **as** **np**

employee = pd.read\_table('path/.tsv')

employee.select\_dtypes(include = [np.number]).dtypes

*#inbuilt methods for data analysis*

employee.describe()

*#number of rows and columns*

employee.shape

employee = pd.read\_table('path/.tsv')

employee.head()

*#concatenating 2 columns*

employee['Name Salary']=employee['Name']+employee['Salary']

employee.head()

*#dropping 2 columns*

employee.drop('Office', axis=1, inplace=**True**)

employee.head()

*#renaming column*

cols =['Name','Posistion', 'Ofc','Age','SDate','Sal']

employee.columns = cols

employee.head()

*#sorting Data*

employee.sort\_values(by='Age', ascending=**True**).head()

employee['Name'].sort\_values()

*#filtering*

employee[employee.Age<25]

employee[(employee.Age<25) & (employee.Name == 'Shou Itou')]

cols = ['Name' , 'Position']

employee[employee.Age<25][cols].head()

employee.head()

employee.mean()

employee.Name.str.upper().head()

employee.Name.str.lower().head()

employee.Position.str.contains('Software')

employee[employee.Position.str.contains('Software')]

employee.head()

employee.Position.str.replace('Engineer','Developer')

employee.Age.min()

employee.Age.max()

employee.groupby('Position').Age.min()

employee.groupby('Position').Age.agg(['count','min','max'])

*#loc - choose number of rows and column*

employee = pd.read\_table('C:/Users/LocalAdmin/Documents/Python/Untitled Folder/DataTablesExample\_TSV.tsv')

*#row 0, all col*

employee.loc[0, :]

*#row 0-2, all col*

employee.loc[0:2, :]

*#row 0-2, 2 col*

employee.loc[0:2, ['Name','Position']]

*#row 0-2, 3 col*

employee.loc[0:2, 'Name':'Office']

*#rows with certain condition*

employee.loc[employee.Position == 'Accountant',:]

*#dropna - drop rows with missing values*

employee = pd.read\_table(‘path/.tsv')

employee.head()

employee.shape

*#drop values with all missing values in any columns*

employee.dropna(how='any').shape

*#drop values with all missing values in specific columns*

employee.dropna(subset=['Name','Salary']).shape

*#fillna*

milkprod = pd.read\_csv('C:/Users/LocalAdmin/Documents/Python/Untitled Folder/MonthlyMilkProduction.csv')

milkprod.head()

milkprod.plot()

*#joins*

*#create dummy data*

dataframe1 = pd.DataFrame({

"employee":["ABC","XYZ","PQR"],

"salary":["100000","125000","80000"]

})

dataframe1

dataframe2 = pd.DataFrame({

"employee":["ABC","XYZ","LMN"],

"salary":["100000","125000","90000"]

})

dataframe2

*#inner join*

dataframe3 = pd.merge(dataframe1,dataframe2, on='employee')

dataframe3

*#outer join*

dataframe3 = pd.merge(dataframe1,dataframe2, on='employee', how='outer')

dataframe3

*#left join*

dataframe3 = pd.merge(dataframe1,dataframe2, on='employee', how='left')

dataframe3

*#right join*

dataframe3 = pd.merge(dataframe1,dataframe2, on='employee', how='right')

dataframe3

*#pivot* webtraffic = pd.read\_csv('C:/Users/LocalAdmin/Documents/Python/Untitled Folder/PivotExample.csv') webtraffic

webtraffic.pivot(index='Page\_name', columns='Date')

*#Pivot table*

webtraffic.pivot\_table(index='Page\_name', aggfunc='sum')

*#Pivot table*

webtraffic.pivot\_table(index='Page\_name', aggfunc='mean')

*#shift*

fb\_finance = pd.read\_csv('C:/Users/LocalAdmin/Documents/Python/Untitled Folder/FB.csv')

fb\_finance.head()

fb\_finance.shift(1).head()

fb\_finance.shift(-1).head()

*#calculate stock price delta*

fb\_finance['previous\_closing\_price']=fb\_finance['Close'].shift(1)

fb\_finance.head()

fb\_finance['previous\_closing\_price'] = fb\_finance['previous\_closing\_price']-fb\_finance['Close']

fb\_finance.head()

fb\_finance['weekly returns'] = (fb\_finance['Close']-fb\_finance['Close'].shift(7))/fb\_finance['Close']\*100

fb\_finance

*#write to csv*

fb\_finance.to\_csv('C:/Users/LocalAdmin/Documents/Python/Untitled Folder/FB.csv')

*#write to csv columns*

fb\_finance.to\_csv('C:/Users/LocalAdmin/Documents/Python/Untitled Folder/FB.csv', columns=['Volume','weekly returns'])