slip1

```
import numpy as np

import pandas as pd

import matplotlib.pyplot as plt


iris = pd.read_csv("Iris.csv")

print (iris.head(10))

iris.plot(kind ="scatter", x ='SepalLengthCm', y ='PetalLengthCm')

plt.grid()
```

---

slip2

```
import numpy as np #used for handling numbers

import pandas as pd #used for handling the dataset

from sklearn.impute import SimpleImputer #used for handling missing data
```

```
dataset = pd.read_csv('Data.csv')

dataset

X = dataset.iloc[:,:-1].values # attributes to determine dependent variable / Class
Y = dataset.iloc[:,-1].values # dependent variable / Class attributes

imputer = SimpleImputer(missing_values=np.nan, strategy='mean')
imputer = imputer.fit(X[:, 1:])

X[:, 1:] = imputer.transform(X[:, 1:])
```

—

slip3

```
import pandas as pd
data = {'name': ['Sheldon', 'Penny', 'Amy', 'Penny', 'Raj', 'Sheldon'],
'episodes': [42, 24, 31, 29, 37, 40],
'gender': ['male', 'female', 'female', 'female', 'male', 'male']}
df = pd.DataFrame(data, columns = ['name','episodes', 'gender'])
print(df)
df_gender = pd.get_dummies(df['gender'])
df_new = pd.concat([df, df_gender], axis=1)
print(df_new)
```

slip4

```
#importing libraries
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

dataset = pd.read_csv('housing_price.csv')
X = dataset.iloc[:, :-1].values #get a copy of dataset exclude last column
y = dataset.iloc[:, 1].values #get array of dataset in column 1st

dataset
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=1/3, random_state=0)
from sklearn.linear_model import LinearRegression
regressor = LinearRegression()
regressor.fit(X_train, y_train)
viz_train = plt
viz_train.scatter(X_train, y_train, color='red')
viz_train.plot(X_train, regressor.predict(X_train), color='blue')
viz_train.title('Salary VS Experience (Training set)')
viz_train.xlabel('Year of Experience')
viz_train.ylabel('Salary')
viz_train.show()
```

```
y_pred = regressor.predict(np.array([2000]).reshape(1, 1))

print(y_pred)
```

—

slip5

```
import numpy as np

import matplotlib.pyplot as plt

import pandas as pd

from sklearn.model_selection import train_test_split

from sklearn.linear_model import LinearRegression

# Importing the dataset

dataset = pd.read_csv('salary_data1.csv')

X = dataset.iloc[:, :-1].values

Y = dataset.iloc[:, 4].values

X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size = 0.2, random_state = 0)

dataset

regressor = LinearRegression()

regressor.fit(X_train, y_train)

x_new = [[5],[2],[1],[2]]

y_pred = regressor.predict(np.array(x_new).reshape(1, 4))

print(y_pred)

accuracy = (regressor.score(X_test,y_test))

print(accuracy)
```

—

slip6

```
#Importing Libraries
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
dataset = pd.read_csv('position_salaries.csv')
X = dataset.iloc[:, 1:2].values
y = dataset.iloc[:, 2].values
dataset

from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=0)

from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import PolynomialFeatures
poly_reg = PolynomialFeatures(degree=4)
X_poly = poly_reg.fit_transform(X)
pol_reg = LinearRegression()
pol_reg.fit(X_poly, y)

def viz_polymonial():
 plt.scatter(X, y, color='red')
 plt.plot(X, pol_reg.predict(poly_reg.fit_transform(X)), color='blue')
 plt.title('Truth or Bluff (Linear Regression)')
 plt.xlabel('Position level')
 plt.ylabel('Salary')
 plt.show()
 return
viz_polymonial()

def viz_polymonial_smooth():
  X_grid = np.arange(min(X), max(X), 0.1)
  X_grid = X_grid.reshape(len(X_grid), 1)

 plt.scatter(X, y, color='red')
 plt.plot(X_grid, pol_reg.predict(poly_reg.fit_transform(X_grid)), color='blue')
 plt.title('Truth or Bluff (Linear Regression)')
 plt.xlabel('Position level')
 plt.ylabel('Salary')
 plt.show()

 return
 viz_polymonial_smooth()

print(pol_reg.predict(poly_reg.fit_transform([[5.5]])))
```

____

slip7

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
```

```
dataset = pd.read_csv("userdata.csv")
dataset

X = dataset.iloc[:, [2, 3]].values
y = dataset.iloc[:, 4].values

from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.25, random_state = 0)

from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)

from sklearn.naive_bayes import GaussianNB
classifier = GaussianNB()
classifier.fit(X_train, y_train)

y_pred = classifier.predict(X_test)

from sklearn.metrics import confusion_matrix

cm = confusion_matrix(y_test, y_pred)
```

```
from matplotlib.colors import ListedColormap

x_set, y_set = X_train, y_train

x1, x2 = np.meshgrid(np.arange(start = x_set[:, 0].min() - 1, stop = x_set[:, 0].max() + 1,

step =0.01),

np.arange(start = x_set[:, 1].min() - 1, stop = x_set[:, 1].max() + 1, step = 0.01))

plt.contourf(x1, x2, classifier.predict(np.array([x1.ravel(),

x2.ravel()]).T).reshape(x1.shape),

alpha = 0.75, cmap = ListedColormap(('purple','green' )))

plt.xlim(x1.min(), x1.max())

plt.ylim(x2.min(), x2.max())

for i, j in enumerate(np.unique(y_set)):

  plt.scatter(x_set[y_set == j, 0], x_set[y_set == j, 1],

      c = ListedColormap(('purple', 'green'))(i), label = j)

plt.title('Decision Tree Algorithm (Training set)')

plt.xlabel('Age')

plt.ylabel('Estimated Salary')

plt.legend()

plt.show()
```

```
from matplotlib.colors import ListedColormap

x_set, y_set = X_test, y_test

x1, x2 = np.meshgrid(np.arange(start = x_set[:, 0].min() - 1, stop = x_set[:, 0].max() + 1,

step =0.01),

np.arange(start = x_set[:, 1].min() - 1, stop = x_set[:, 1].max() + 1, step = 0.01))

plt.contourf(x1, x2, classifier.predict(np.array([x1.ravel(),
x2.ravel()]).T).reshape(x1.shape),
alpha = 0.75, cmap = ListedColormap(('purple','green' )))
plt.xlim(x1.min(), x1.max())
plt.ylim(x2.min(), x2.max())
for i, j in enumerate(np.unique(y_set)):
  plt.scatter(x_set[y_set == j, 0], x_set[y_set == j, 1],
      c = ListedColormap(('purple', 'green'))(i), label = j)
plt.title('Decision Tree Algorithm (Test set)')
plt.xlabel('Age')
plt.ylabel('Estimated Salary')
plt.legend()
plt.show()
```

---

slip8

```
import numpy as np
import pandas as pd
dataset = pd.read_csv("play_tennis .csv")
dataset

from sklearn.preprocessing import LabelEncoder
Le = LabelEncoder()
dataset['outlook'] = Le.fit_transform(dataset['outlook'])
dataset['temp'] = Le.fit_transform(dataset['temp'])
dataset['humidity'] = Le.fit_transform(dataset['humidity'])
dataset['wind'] = Le.fit_transform(dataset['wind'])
```

```
dataset['play'] = Le.fit_transform(dataset['play'])
X = dataset.iloc[:, :-1].values
Y = dataset.iloc[:, 4].values

from sklearn import tree
clf = tree.DecisionTreeClassifier(criterion = 'entropy')
clf = clf.fit(X, Y)

tree.plot_tree(clf)
X_pred = clf.predict(X)
X_pred==Y
```

—

slip9

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

dataset = pd.read_csv("Social_Network_Ads.csv")
dataset

X = dataset.iloc[:, [2, 3]].values
y = dataset.iloc[:, 4].values

from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.25, random_state =
0)

from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)

from sklearn.svm import SVC
classifier = SVC(kernel='linear', random_state = 0)
classifier.fit(X_train, y_train)

y_pred = classifier.predict(X_test)

from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred)

from matplotlib.colors import ListedColormap
x_set, y_set = X_train, y_train
x1, x2 = np.meshgrid(np.arange(start = x_set[:, 0].min() - 1, stop = x_set[:, 0].max() + 1,
step =0.01),
np.arange(start = x_set[:, 1].min() - 1, stop = x_set[:, 1].max() + 1, step = 0.01))
plt.contourf(x1, x2, classifier.predict(np.array([x1.ravel(),
x2.ravel()]).T).reshape(x1.shape),
alpha = 0.75, cmap = ListedColormap(('red','green' )))
plt.xlim(x1.min(), x1.max())
plt.ylim(x2.min(), x2.max())

for i, j in enumerate(np.unique(y_set)):

  plt.scatter(x_set[y_set == j, 0], x_set[y_set == j, 1],

      c = ListedColormap(('red', 'green'))(i), label = j)

plt.title('SVM classifier (Training set)')

plt.xlabel('Age')

plt.ylabel('Estimated Salary')

plt.legend()

plt.show()
```

```
from matplotlib.colors import ListedColormap

x_set, y_set = X_test, y_test

x1, x2 = np.meshgrid(np.arange(start = x_set[:, 0].min() - 1, stop = x_set[:, 0].max() + 1,

step =0.01),

np.arange(start = x_set[:, 1].min() - 1, stop = x_set[:, 1].max() + 1, step = 0.01))

plt.contourf(x1, x2, classifier.predict(np.array([x1.ravel(),

x2.ravel()]).T).reshape(x1.shape),

alpha = 0.75, cmap = ListedColormap(('red','green' )))

plt.xlim(x1.min(), x1.max())

plt.ylim(x2.min(), x2.max())

for i, j in enumerate(np.unique(y_set)):

  plt.scatter(x_set[y_set == j, 0], x_set[y_set == j, 1],

      c = ListedColormap(('red', 'green'))(i), label = j)

plt.title('SVM classifier (Test set)')

plt.xlabel('Age')
plt.ylabel('Estimated Salary')
plt.legend()
plt.show()
```

____
__

slip10

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

iris = pd.read_csv("Iris.csv")
print (iris.head(10))
iris.plot(kind ="scatter", x ='SepalLengthCm', y ='PetalLengthCm')
plt.grid()
```