1.

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
dataset = pd.read_csv("suv_data.csv")
dataset.head()
x = dataset.iloc[:, [2, 3]].values
y = dataset.iloc[:, 4].values
bool_series=pd.isnull(dataset["Gender"])
dataset[bool_series]
bool_series=pd.notnull(dataset["Gender"])
dataset[bool_series] dataset[10:25]
new_data=dataset.dropna(axis=0,how='any')
new_data
dataset.replace(to_replace=np.nan,value=-99)
dataset["Gender"].fillna("No
Gender",inplace=True)
dataset
print("Old data frame length:", len(dataset))
print("New data frame length:", len(dataset))
print("Number of rows with at least 1 NA
value:",
len(dataset)-len(new_data))
Old data frame length: 400
New data frame length: 400
Number of rows with at least 1 NA value: 0
new_df1=dataset.fillna(method="ffill")
new_df1
new_df3=dataset.dropna(how='all')
new_df3
```

2.

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
data = pd.read_csv('Titanic-Dataset.csv')
print(data)
x = data.drop('Survived', axis = 1)
y = data['Survived']
print(x)
print(y)
x.drop(['Name', 'Ticket', 'Cabin'],axis = 1, inplace
=
True)
print(x)
x['Age'] = x['Age'].fillna(x['Age'].mean())
print(x)
x['Embarked'] =
x['Embarked'].fillna(x['Embarked'].mode()[0])
print(x)
x = pd.get_dummies(x, columns = ['Sex',
'Embarked'],prefix = ['Sex',
'Embarked'],drop_first =
True)
print(x)
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)
print(x_train)
print(y_train)
from sklearn.preprocessing import
StandardScaler
std_x = StandardScaler()
x_train = std_x.fit_transform(x_train)
x_test = std_x.transform(x_test)
print(x_train)
```

3.

```
from pandas import read_csv
from numpy import set_printoptions
from sklearn.model_selection import
train_test_split
from sklearn.feature_selection import
SelectKBest
from sklearn.feature_selection import f_classif
from matplotlib import pyplot
path=r'diabetes.csv'
names=['preg','plas','pres','skin','test','mass','pe
ds','ag
e','class']
dataframe=read_csv(path,names=names)
dataframe.head()
array=dataframe.values
x=array[:,0:8]
y=array[:,8]
print(x)
print(y)
x_train,x_test,y_train,y_test,=train_test_split(x,y
,test_
size=0.33,random_state=1)
fs= SelectKBest(score_func=f_classif,k='all')
fs.fit(x_train,y_train)
x_train_fs=fs.transform(x_train)
x_test_fs=fs.transform(x_test)
for i in range(len(fs.scores_)):
print('feature %d:%f'%(i,fs.scores_[i]))
pyplot.bar([i for i in
range(len(fs.scores_))],fs.scores_)
pyplot.show()
```

4.

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns from
sklearn.feature_selection
import chi2
df=pd.read_csv('loandata.csv')
df.head()
from sklearn.preprocessing import LabelEncoder
for col in df.columns:
 le=LabelEncoder()
 df[col]=le.fit_transform(df[col])
df.head()
x=df.iloc[:,0:6]
y=df.iloc[:,-1]
f_score=chi2(x,y)
f_score
p_value=pd.Series(f_score[1], index=x.columns)
p_value.sort_values(ascending=False,inplace=Tr
ue)
p_value
p_value.plot(kind="bar")
plt.xlabel("Features", fontsize=20)
plt.ylabel("p_values", fontsize=20)
plt.title("chi squared test base on p value")
plt.show()
```

5

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns
from sklearn import tree
from sklearn import metrics
from sklearn.metrics import accuracy_score,
classification_report
from sklearn.datasets import load_iris
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import
train_test_split
iris = load_iris()
iris = sns.load_dataset('iris')
iris.head()
x=iris.iloc[:,:-1]
y=iris.iloc[:,-1]
x_train, x_test, y_train, y_test =
train_test_split(x,y, test_size=0.33,
random_state=42)
treemodel = DecisionTreeClassifier()
treemodel.fit(x_train, y_train)
y_pred = treemodel.predict(x_test)
plt.figure(figsize=(20,30))
tree.plot_tree(treemodel, filled=True)
print(classification_report(y_test, y_pred))
from sklearn.metrics import confusion_matrix
cm=confusion_matrix(y_test, y_pred)
print("Confusion Matrix:")
print(cm)
from sklearn.metrics import accuracy_score
accuracy_score(y_test, y_pred)
```

6.

```
import numpy as nm
import matplotlib.pyplot as mtp
import pandas as pd
datasetpd.read_csv('User_data.csv")
x= dataset.iloc[:,[2,3]].values
print(x) from sklearn.preprocessing import
StandardScaler from sklearn.naive bayes import
GaussianNB
y=dataset.iloc[:,4].values
print(y)
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)
sc StandardScaler()
x_train sc.fit_transform(x_train)
x_testsc.fit_transform(x_test)
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)
print("Confusion Matrix:")
print(cm)
```

7.

```
 import numpy as np
model's performance.
import matplotlib.pyplot as plt
import pandas as pd
dataset pd.read_csv('Social Network Ads.csv')
X dataset.iloc[:, :-1].values
ydataset.iloc[:,-1].values
dataset.head()
from sklearn.model_selection import
train_test_split
X_train, X test, y train, y_test train_test_split(X,
y, test
size 0.20, random state-42)
X
y
from sklearn.preprocessing import
StandardScaler
se StandardScaler()
X_train sc.fit_transform(X_train)
X_test sc.transform(X_test)
print(X_train)
from sklearn.neighbors import KNeighbors
Classifier
classifier KNeighborsClassifier(n_neighbors 5,
metrie
'minkowski', p2)
classifier.fit(X_train, y_train)
print(classifier.predict(sc.transform([[46,28000]])
))
y_pred classifier.predict(X_test)
print(np.concatenate((y_pred.reshape(len(y_pre
d), 1),
y_test.reshape(len(y_test),1)),1))
from sklearn.metrics import confusion_matrix,
accuracy_score
cm confusion_matrix(y_test, y_pred)
print(cm)
accuracy_score(y_test, y_pred)
```

8.

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
datasetpd.read_csv('Salary_Data.csv')
x=dataset.iloc[:, :-1],values
y=dataset.iloc[:,-1].values
dataset.head()
from sklearn.model_selection import
train_test_split
x_train,x_test,y_train,y_testtrain_test_split(x,y,t
est_size=1/3, random_state-0)
print(x_train)
print(x_test)
print(y train)
print(y_test)
from sklearn.linear model import
LinearRegression
regressor-LinearRegression()
regressor.fit(x_train.y_train)
y_pred-regressor.predict(x_test)
print(y_test)
print(y_pred)
print(np.concatenate((y_test.reshape(len(y_test)
,1),y_
pred.reshape(len(y_pred),131,13)
from sklearn.metrics import
mean_squared_error
mean mean_squared_error(y_test,y_pred)
mean
plt.scatter(x_train,y_train,color='red')
plt.plot(x train
regressor.predict(x_train),color="blue")
plt.title('salary vs Experience(Training set)')
plt.xlabel('years of Experience')
plt.ylabel('salary)
plt.show()
plt.scatterfx test, y test, color-'red')
plt.plot(x train, regressor.predict(x_train),
color="blue")
plt.titlet Salary vs Experience (Test set)')
plt.xlabel("Years of Experience')
plt.ylabel("Salary)
plt.show()
```

9.

```
import numpy as np
import pandas as pd
from sklearn.model_selection import
train_test_split
from sklearn.linear_model import
LinearRegression
from sklearn.metrics import
mean_squared_error,
r2_score
from sklearn.preprocessing import
OneHotEncoder
from sklearn.compose import
ColumnTransformer
data = pd.read_csv('50_Startups.csv')
df = pd.DataFrame(data)
print("Dataset:")
print(df.head())
X = df.drop(columns=['Profit'])
y = df['Profit']
column_transformer = ColumnTransformer(
 transformers=[('encoder', OneHotEncoder(),
['State'])], remainder='passthrough')
X = column_transformer.fit_transform(X)
X_train, X_test, y_train, y_test =
train_test_split(X, y,
test_size=0.2, random_state=42)
model = LinearRegression()
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)
print(f"Mean Squared Error: {mse}")
print(f"R-squared Score: {r2}")
print("Coefficients:", model.coef_)
print("Intercept:", model.intercept_)
```

10.

```
import numpy as nm
import matplotlib.pyplot as mtp
import pandas as pd
dataset = pd.read_csv('Mall_Customers.csv')
x = dataset.iloc[:, [3, 4]].values
dataset.head()
import scipy.cluster.hierarchy as shc
dendro = shc.dendrogram(shc.linkage(x,
method="ward"))
mtp.title("Dendrogrma Plot")
mtp.ylabel("Euclidean Distances")
mtp.xlabel("Customers")
mtp.show()
from sklearn.cluster import
AgglomerativeClustering
hc = AgglomerativeClustering(n_clusters=5,
metric='euclidean', linkage='ward')
y_pred = hc.fit_predict(x)
#visulaizing the clusters
mtp.scatter(x[y_pred == 0, 0], x[y_pred == 0, 1], s
=100, c = 'blue', label = 'Cluster 1')
mtp.scatter(x[y_pred == 1, 0], x[y_pred == 1, 1], s
=100, c = 'green', label = 'Cluster 2')
mtp.scatter(x[y_pred== 2, 0], x[y_pred == 2, 1], s
=100, c = 'red', label = 'Cluster 3')
mtp.scatter(x[y_pred == 3, 0], x[y_pred == 3, 1], s
=100, c = 'cyan', label = 'Cluster 4')
mtp.scatter(x[y_pred == 4, 0], x[y_pred == 4, 1], s
=100, c = 'magenta', label = 'Cluster 5')
mtp.title('Clusters of customers')
mtp.xlabel('Annual Income (k$)')
mtp.ylabel('Spending Score (1-100)')
mtp.legend()
mtp.show()
```