ARTIFICIAL INTELLIGENCE LAB
(CSL5402)

Name: Lakhan Kumawat

Roll: 1906055

Program: B.Tech CSE
(5th Sem JUL-DEC 2021)

Small project based AI Lab Assignments I & II

Name : Lakhan Kumawat Roll No : 1906055 Course : CSL5402

ASSIGNMENT - I

Supervised Machine Learning (Linear Regression, Logistic Regression, and Support Vector Machine):

1. Read breast-cancer dataset (shared with the named "breast_cancer.csv" in the file section inside the dataset folder) using Linear Regression, Logistic Regression, and SVM.

2. Read and analyze the breast_cancer dataset (shared with the named "breast_cancer.csv" in the file section inside the dataset folder) using Linear Regression, Logistic Regression, and SVM, calculate the Precision, Recall, Accuracy and F1-score.

3. Display the graph plots of the Scatter and histogram using Linear Regression, and SVM methods. Also display the boxplot and scatter plots using Logistic Regression method.

Program Code:

```
import numpy as np
import pandas as pd

# READING THE DATASET
df = pd.read_csv('breast_cancer.csv')
df.head()
```

OUTPUT:

	id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	sm
0	842302	M	17.99	10.38	122.80	1001.0	
1	842517	M	20.57	17.77	132.90	1326.0	
2	84300903	M	19.69	21.25	130.00	1203.0	
3	84348301	M	11.42	20.38	77.58	386.1	
4	84358402	M	20.29	14.34	135.10	1297.0	

```
#Plot 11
```

```
df.shape
```

```
(569, 33)
```

```
df.isna().sum()
```

```
id                         0
diagnosis                  0
radius_mean                0
texture_mean               0
perimeter_mean             0
area_mean                  0
smoothness_mean            0
compactness_mean           0
concavity_mean             0
concave points_mean        0
symmetry_mean              0
fractal_dimension_mean     0
radius_se                  0
texture_se                 0
perimeter_se               0
area_se                    0
smoothness_se              0
compactness_se             0
concavity_se               0
concave points_se          0
symmetry_se                0
fractal_dimension_se       0
radius_worst               0
texture_worst              0
perimeter_worst            0
area_worst                 0
```

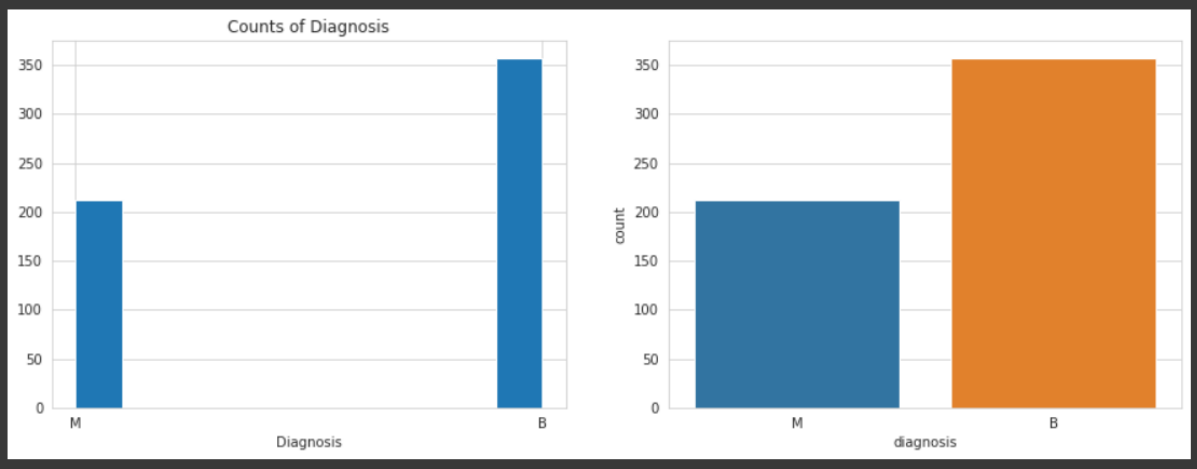
```
smoothness_worst             0
compactness_worst            0
concavity_worst              0
concave points_worst         0
symmetry_worst               0
fractal_dimension_worst      0
Unnamed: 32                569
dtype: int64
```

```
df = df.dropna(axis=1)

df['diagnosis'].value_counts()

#DATA VISUALISATION

import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px

%matplotlib inline
sns.set_style('whitegrid')

plt.figure(figsize=(15, 5))

plt.subplot(1, 2, 1)
plt.hist( df.diagnosis)

plt.title("Counts of Diagnosis")
plt.xlabel("Diagnosis")


plt.subplot(1, 2, 2)

sns.countplot('diagnosis', data=df); #Plot 12
```

OUTPUT:

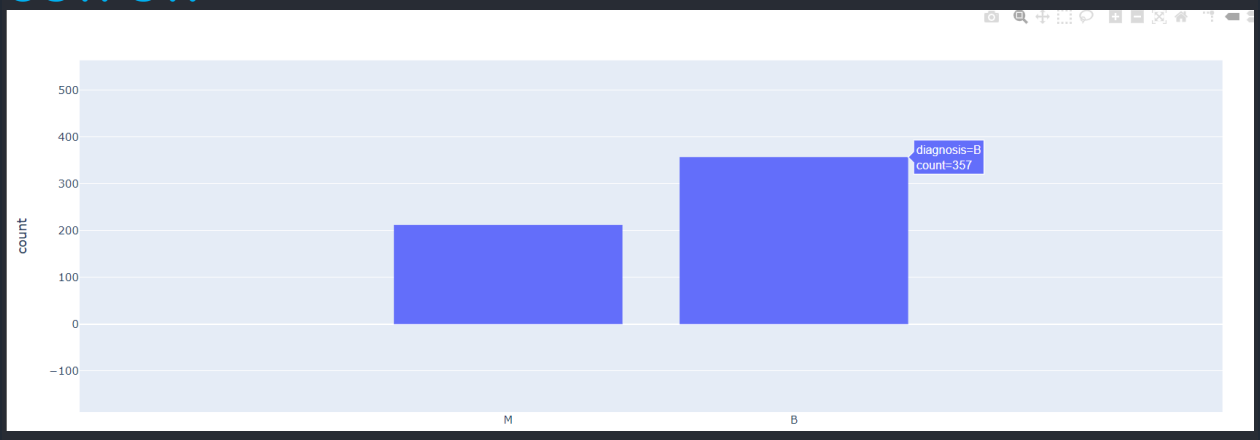
Name : Lakhan Kumawat Roll No : 1906055 Course : CSL5402

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning:

Pass the following variable as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, a
```




```
px.histogram(df, x='diagnosis')  #Plot 13
```

OUTPUT:




```
"""# **Multiple Linear Regression**"""

from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import MinMaxScaler

"""* Encoding M->1 and B->0 """

from sklearn.preprocessing import LabelEncoder
labelencoder_Y = LabelEncoder()
```

```
df.iloc[:,1] =
labelencoder_Y.fit_transform(df.iloc[:,1].values)

df.head()
```

OUTPUT:

	id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean
0	842302	1	17.99	10.38	122.80	1001.0	0.11840
1	842517	1	20.57	17.77	132.90	1326.0	0.08474
2	84300903	1	19.69	21.25	130.00	1203.0	0.10960
3	84348301	1	11.42	20.38	77.58	386.1	0.14250
4	84358402	1	20.29	14.34	135.10	1297.0	0.10030

```
"""* Split the dataset into independent (X) and dependent (Y)
data sets"""

X = df.iloc[:,2:31].values
Y = df.iloc[:,1].values

type(df)
```

```
pandas.core.frame.DataFrame
```

```
"""* Split the data set into 75% training and 25% testing"""

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)

"""* Scale the data (feature scaling)"""

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

```
"""* Linear Model"""

linear_model = LinearRegression()
linear_model.fit(X_train, Y_train)
```

```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None,
normalize=False)
```

```
"""* Predictions"""

predictions = linear_model.predict(X_test)

predictions = np.array(list(map(lambda x : 1 if(x>0.5) else
0,predictions)))
from sklearn.metrics import classification_report ,
confusion_matrix
print(confusion_matrix(Y_test,predictions))
```

```
[[90  0]
 [ 8 45]]
```

```
"""* Accuracy Description"""

print(classification_report(Y_test,predictions))
```

```
              precision    recall  f1-score   support

           0       0.97      0.96      0.96        90
           1       0.93      0.94      0.93        53

    accuracy                           0.95       143
   macro avg       0.95      0.95      0.95       143
weighted avg       0.95      0.95      0.95       143
```

```
"""# **Logistic Regression**"""

from sklearn.linear_model import LogisticRegression

log_model = LogisticRegression(random_state=0)
```

```
log_model.fit(X_train, Y_train)
```

```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, l1_ratio=None, max_iter=100,
                   multi_class='auto', n_jobs=None, penalty='l2',
                   random_state=0, solver='lbfgs', tol=0.0001, verbose=0,
                   warm_start=False)
```

```
"""* Predictions"""

predictions = log_model.predict(X_test)

from sklearn.metrics import classification_report ,
confusion_matrix
print(confusion_matrix(Y_test,predictions))
```

```
[[86  4]
 [ 3 50]]
```

```
"""* Accuracy Description"""

print(classification_report(Y_test,predictions))


from sklearn.metrics import accuracy_score
print(accuracy_score(Y_test, log_model.predict(X_test)))
```

```
              precision    recall  f1-score   support

           0       0.97      0.96      0.96        90
           1       0.93      0.94      0.93        53

    accuracy                           0.95       143
   macro avg       0.95      0.95      0.95       143
weighted avg       0.95      0.95      0.95       143

0.951048951048951
```

```
"""# **SVM**"""
```

```
from sklearn.svm import SVC

classifier = SVC(kernel='linear', random_state=0)
classifier.fit(X_train, Y_train)
```

```
SVC(C=1.0, break_ties=False, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma='scale',
kernel='linear',
    max_iter=-1, probability=False, random_state=0, shrinking=True,
tol=0.001,
    verbose=False)
```

```
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(Y_test, classifier.predict(X_test))
print(cm)
```

```
[[86  4]
 [ 2 51]]
```

```
from sklearn.metrics import classification_report
print(classification_report(Y_test,
classifier.predict(X_test)))

from sklearn.metrics import accuracy_score
print(accuracy_score(Y_test, classifier.predict(X_test)))
```

```
              precision    recall  f1-score   support

           0       0.98      0.96      0.97        90
           1       0.93      0.96      0.94        53

    accuracy                           0.96       143
   macro avg       0.95      0.96      0.96       143
weighted avg       0.96      0.96      0.96       143

0.958041958041958
```

Name : Lakhan Kumawat Roll No : 1906055 Course : CSL5402

ASSIGNMENT – II

Un-Supervised Machine Learning (K-means Clustering, KNN)

1. Read heart dataset and adult dataset (shared with the named “heart.csv” and “adult.csv”
in the file section inside the dataset folder) using KNN (heart dataset) K-means (adult
dataset).

2. Read and analyze the heart dataset and adult dataset (shared with the named “heart.csv”
and “adult.csv” in the file section inside the dataset folder) using KNN (heart dataset)
K-means (adult dataset) calculate the Accuracy, Precision, Recall, and F1-score.

3. Plot the scatter and elbow point graphs by using KNN and scatter graph using K-means
methods.

Program Code:

```
# **KNN**
"""* Importing libraries"""

import pandas as pd
import numpy as np
"""* Read the dataset"""
```

```
df = pd.read_csv('heart.csv')
df.head()
```

OUTPUT:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
0	63	1	3	145	233	1	0	150	0	2.3	0	0	1	1
1	37	1	2	130	250	0	1	187	0	3.5	0	0	2	1
2	41	0	1	130	204	0	0	172	0	1.4	2	0	2	1
3	56	1	1	120	236	0	1	178	0	0.8	2	0	2	1
4	57	0	0	120	354	0	1	163	1	0.6	2	0	2	1

```
df.shape
```

```
(303, 14)
```

```
df.describe()
```

OUTPUT:

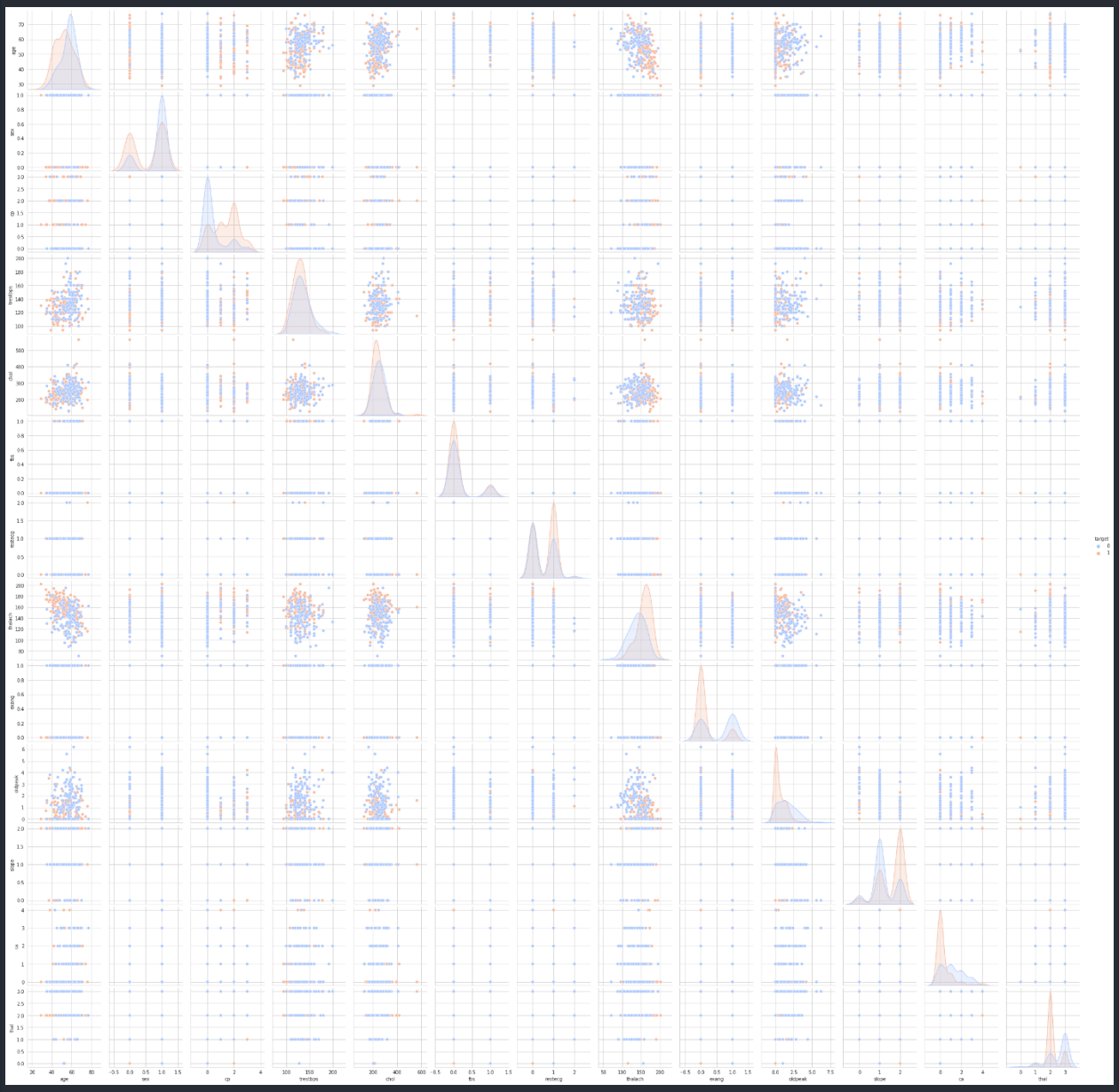
	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
count	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000
mean	54.366337	0.683168	0.966997	131.623762	246.264026	0.148515	0.528053	149.646865	0.326733	1.039604	1.399340	0.729373	2.313531	0.544554
std	9.082101	0.466011	1.032052	17.538143	51.830751	0.356198	0.525860	22.905161	0.469794	1.161075	0.616226	1.022606	0.612277	0.498835
min	29.000000	0.000000	0.000000	94.000000	126.000000	0.000000	0.000000	71.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	47.500000	0.000000	0.000000	120.000000	211.000000	0.000000	0.000000	133.500000	0.000000	0.000000	1.000000	0.000000	2.000000	0.000000
50%	55.000000	1.000000	1.000000	130.000000	240.000000	0.000000	1.000000	153.000000	0.000000	0.800000	1.000000	0.000000	2.000000	1.000000
75%	61.000000	1.000000	2.000000	140.000000	274.500000	0.000000	1.000000	166.000000	1.000000	1.600000	2.000000	1.000000	3.000000	1.000000
max	77.000000	1.000000	3.000000	200.000000	564.000000	1.000000	2.000000	202.000000	1.000000	6.200000	2.000000	4.000000	3.000000	1.000000

```
"""* Data Visualization"""

# Commented out IPython magic to ensure Python compatibility.
import matplotlib.pyplot as plt
import seaborn as sns

# %matplotlib inline
```

Name : Lakhan Kumawat Roll No : 1906055 Course : CSL5402

```
sns.set_style('whitegrid')
plt.figure(figsize=(30, 15))
sns.pairplot(df, hue='target', palette='coolwarm')
```

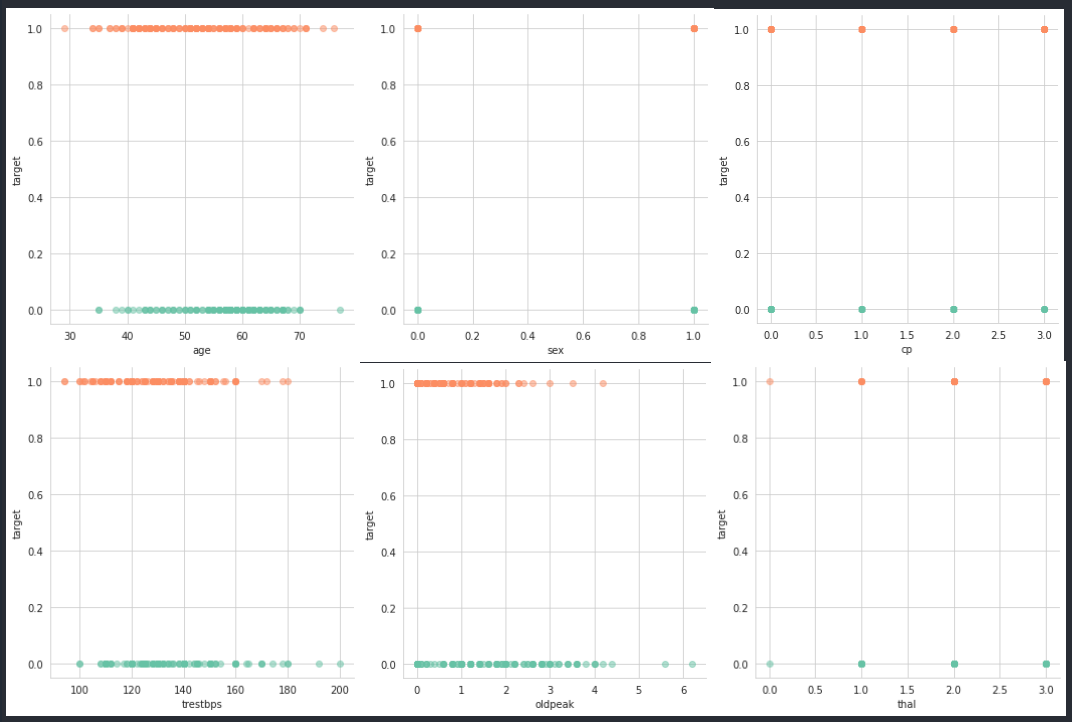
OUTPUT:



```
for col in df.drop(['target'], axis=1).columns:
    sns.lmplot(data=df,
```

```
            x=col,
            y='target',
            fit_reg=False,
            hue='target',
            palette='Set2',
            legend=False,
            scatter_kws={'alpha': 0.5})
```

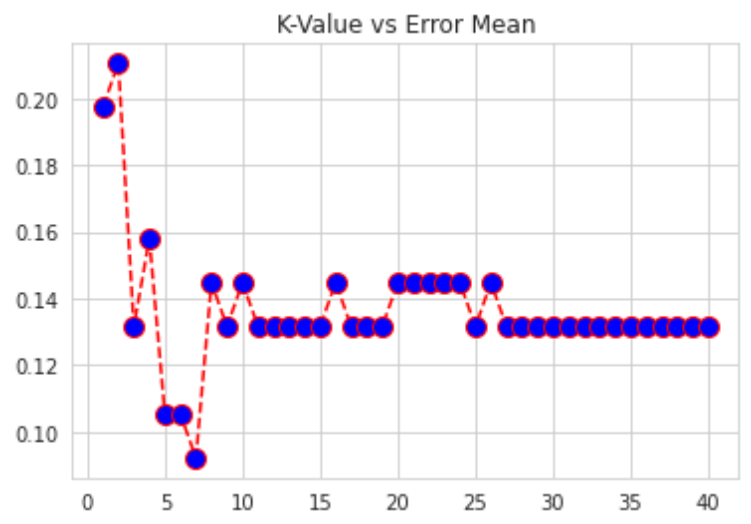
OUTPUT:




```
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import train_test_split
"""* Scaling and Splitting the data into training and testing
data"""

X = df.drop(['target'], axis=1)
Y = df['target']

from sklearn.preprocessing import StandardScaler
```

```
stds = StandardScaler()
X = stds.fit_transform(X)

X_train, X_test, Y_train, Y_test = train_test_split(X,
                                                    Y,
                                                    test_size=0
.25,
                                                    random_stat
e=42)
"""* Calculating the value of **k**"""


def knnprediction(k, X_train, Y_train, X_test):
    knn = KNeighborsClassifier(n_neighbors=k)
    knn.fit(X_train, Y_train)
    pred = knn.predict(X_test)
    return pred


error_mean = []
for k in range(1, 41):
    pred_i = knnprediction(k, X_train, Y_train, X_test)
    error_mean.append(np.mean(pred_i != Y_test))

plt.plot(range(1, 41),
         error_mean,
         linestyle='dashed',
         color='r',
         marker='o',
         markerfacecolor='b',
         markersize=10)
plt.title('K-Value vs Error Mean')
plt.show()
```

OUTPUT:




```
"""* Value of K = 7"""

predictions = knnprediction(7, X_train, Y_train, X_test)
"""* Accuracy Description"""

from sklearn.metrics import classification_report,
confusion_matrix

cm = confusion_matrix(Y_test, predictions)
print(cm)
```

```
[[31  4]
 [ 3 38]]
```

```
print(classification_report(Y_test, predictions))
```

```
              precision    recall  f1-score   support

           0       0.91      0.89      0.90        35
           1       0.90      0.93      0.92        41
    accuracy                           0.91        76
   macro avg       0.91      0.91      0.91        76
weighted avg       0.91      0.91      0.91        76
```

```
"""# **K-means Clustering**
```

Name : Lakhan Kumawat Roll No : 1906055 Course : CSL5402

```
* Importing Libraries
"""

import pandas as pd
import numpy as np
"""* Reading the dataset"""

dfs = pd.read_csv('adult.csv', skipinitialspace=True)
dfs.head()
```

OUTPUT:

	39	State-gov	77516	Bachelors	13	Never-married	Adm-clerical	Not-in-family	White	Male	2174	0	40	United-States	<=50K
0	50	Self-emp-not-inc	83311	Bachelors	13	Married-civ-spouse	Exec-managerial	Husband	White	Male	0	0	13	United-States	<=50K
1	38	Private	215646	HS-grad	9	Divorced	Handlers-cleaners	Not-in-family	White	Male	0	0	40	United-States	<=50K
2	53	Private	234721	11th	7	Married-civ-spouse	Handlers-cleaners	Husband	Black	Male	0	0	40	United-States	<=50K
3	28	Private	338409	Bachelors	13	Married-civ-spouse	Prof-specialty	Wife	Black	Female	0	0	40	Cuba	<=50K
4	37	Private	284582	Masters	14	Married-civ-spouse	Exec-managerial	Wife	White	Female	0	0	40	United-States	<=50K

```
"""* Labelling the column"""

dfs.columns = [
    'age', 'workclass', 'fnlwgt', 'education', 'education-num',
    'marital-status', 'occupation', 'relationship', 'race',
'gender',
    'captialgain', 'capitalloss', 'hoursperweek', 'native-
country', 'class'
]
dfs.head()
```

OUTPUT:

	age	workclass	fnlwgt	education	education-num	marital-status	occupation	relationship	r
0	50	Self-emp-not-inc	83311	Bachelors	13	Married-civ-spouse	Exec-managerial	Husband	W
1	38	Private	215646	HS-grad	9	Divorced	Handlers-cleaners	Not-in-family	W
2	53	Private	234721	11th	7	Married-civ-spouse	Handlers-cleaners	Husband	Bl
3	28	Private	338409	Bachelors	13	Married-civ-spouse	Prof-specialty	Wife	Bl
4	37	Private	284582	Masters	14	Married-civ-spouse	Exec-managerial	Wife	W

```
dfs.describe()
```

OUTPUT:

	age	fnlwgt	education-num	captialgain	capitalloss	hoursperweek
count	32560.000000	3.256000e+04	32560.000000	32560.000000	32560.000000	32560.000000
mean	38.581634	1.897818e+05	10.080590	1077.615172	87.306511	40.437469
std	13.640642	1.055498e+05	2.572709	7385.402999	402.966116	12.347618
min	17.000000	1.228500e+04	1.000000	0.000000	0.000000	1.000000
25%	28.000000	1.178315e+05	9.000000	0.000000	0.000000	40.000000
50%	37.000000	1.783630e+05	10.000000	0.000000	0.000000	40.000000
75%	48.000000	2.370545e+05	12.000000	0.000000	0.000000	45.000000
max	90.000000	1.484705e+06	16.000000	99999.000000	4356.000000	99.000000

```
dfs.dtypes
```

```
age                int64
workclass         object
fnlwgt             int64
education         object
education-num      int64
marital-status    object
occupation        object
relationship      object
race              object
gender            object
captialgain        int64
capitalloss        int64
hoursperweek       int64
native-country    object
class             object
```

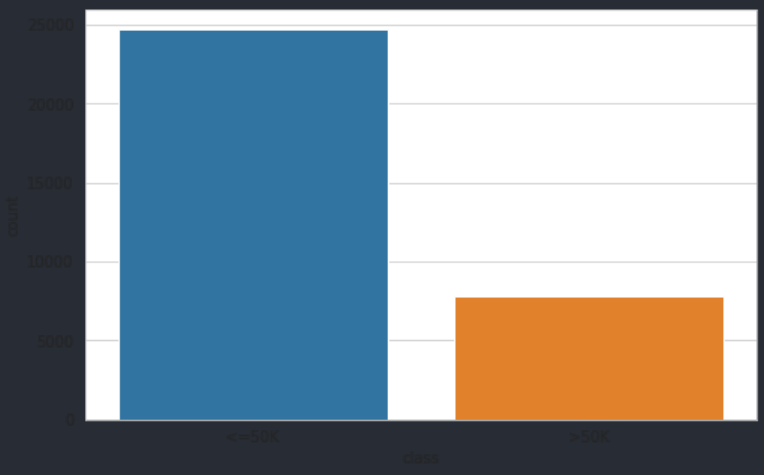
```
dtype: object
```

```
dfs.shape
```

```
(32560, 15)
```

```
"""* Data Visualization"""

# Commented out IPython magic to ensure Python compatibility.
import seaborn as sns
import matplotlib.pyplot as plt
# %matplotlib inline

plt.figure(figsize=(8, 5))
sns.countplot(x="class", data=dfs)
```

OUTPUT:




```
from sklearn.preprocessing import MinMaxScaler
from sklearn.model_selection import train_test_split

from sklearn.cluster import KMeans

from sklearn.metrics import classification_report,
confusion_matrix, accuracy_score
```

```
df_encoded = pd.get_dummies(
    data=dfs,
    columns=dfs.select_dtypes(include=[object]).columns,
    drop_first=True)
df_encoded.head()
```

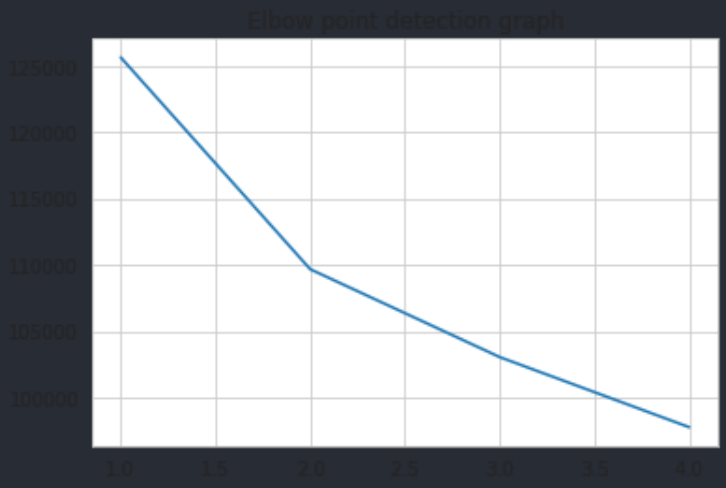
OUTPUT:

	age	fnlwgt	education-num	captialgain	capitalloss	hoursperweek	workclass_Federal-gov	workclass_Local-gov	workclass_Ne wo
0	50	83311	13	0	0	13	0	0	
1	38	215646	9	0	0	40	0	0	
2	53	234721	7	0	0	40	0	0	
3	28	338409	13	0	0	40	0	0	
4	37	284582	14	0	0	40	0	0	

5 rows × 101 columns

```
X = df_encoded.drop('class_>50K', axis=1)
Y = df_encoded['class_>50K']

X_scaled = MinMaxScaler().fit_transform(X)
"""* Elbow point Graph"""

inertia = []
for k in range(1, 5):
    k_cluster = KMeans(n_clusters=k)
    k_cluster.fit_transform(X_scaled)
    inertia.append(k_cluster.inertia_)

plt.plot(range(1, 5), inertia)
plt.title("Elbow point detection graph")
plt.show()
```

OUTPUT:




```
K_model = KMeans(n_clusters=2, random_state=101)
K_model.fit(X)
```

```
KMeans(algorithm='auto', copy_x=True, init='k-means++', max_iter=300,
       n_clusters=2, n_init=10, n_jobs=None, precompute_distances='auto',
       random_state=101, tol=0.0001, verbose=0)
```

```
prediction = K_model.labels_
prediction
```

```
array([0, 0, 0, ..., 0, 0, 1], dtype=int32)
```

```
Y.values
```

```
array([0, 0, 0, ..., 0, 0, 1], dtype=uint8)
```

```
"""* Accuracy Description"""
```

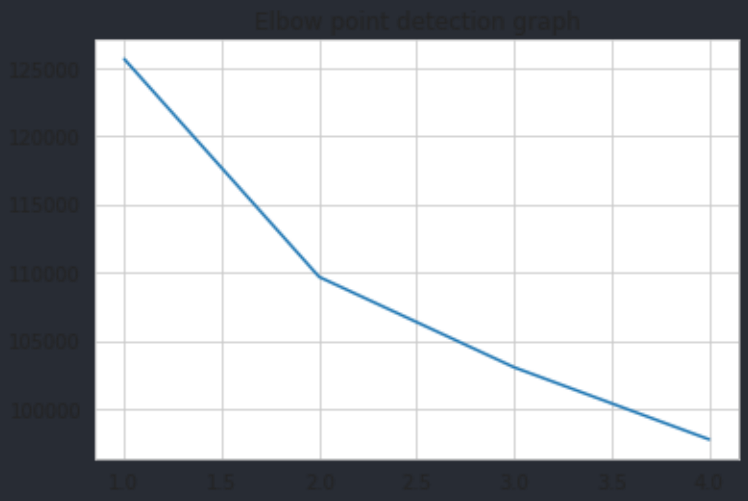
```
print(classification_report(Y, prediction))
print(confusion_matrix(Y, prediction))
```

```
              precision    recall  f1-score   support

           0       0.75      0.74      0.75     24719
           1       0.23      0.24      0.23      7841

    accuracy                           0.62     32560
   macro avg       0.49      0.49      0.49     32560
weighted avg       0.63      0.62      0.62     32560

[[18275  6444]
```

```
 [ 5948  1893]]
```

```
plt.plot(range(1, 5), inertia)
plt.title("Elbow point detection graph")
plt.show()
```

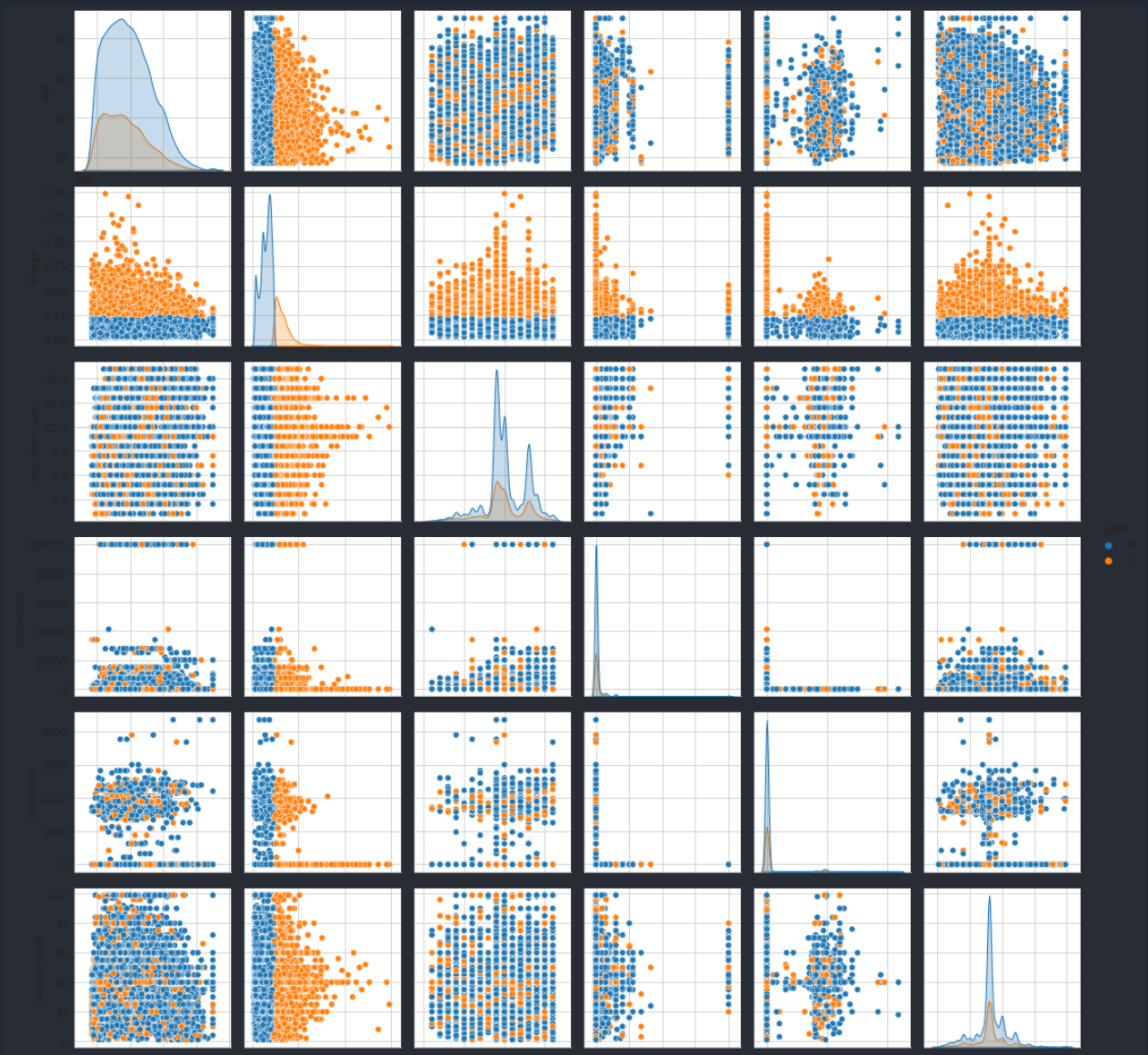
OUTPUT:




```
num_cols = dfs._get_numeric_data().columns

dfs['pred'] = prediction

sns.pairplot(data=dfs, vars=num_cols, hue='pred')
```

OUTPUT:

Name : Lakhan Kumawat Roll No : 1906055 Course : CSL5402



End Of Assignment