Lab Manual

Subject:	**Machine Learning**
Course Code:	**AI-414**
Muneeb Sajid	**2023-BS-AI-028**
Supervised By:	**Mr. Saeed**

DEPARTMENT OF COMPUTATIONAL SCIENCES FACULTY OF INFORMATION TECHNOLOGY

The University of Faisalabad

Table of Content

Project No 01:

Car Price Prediction

Import Drive

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
```

Importing the Dependencies:

```
import pandas as pd
import numpy as np
import sklearn
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import StandardScaler, LabelEncoder
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score
```

Data Collection and Data Processing

```
data = pd.read_csv('/content/drive/MyDrive/Project/car_price_prediction.csv')
```

Data.head

```
data.head()
```

	car_ID	symboling	CarName	fueltype	aspiration	doornumber	carbody	drivewheel	enginelocation	wheelbase	...	enginesize	fuelsystem	boreratio	stro
0	1	3	alfa-romero giulia	gas	std	two	convertible	rwd	front	88.6	...	130	mpfi	3.47	2
1	2	3	alfa-romero stelvio	gas	std	two	convertible	rwd	front	88.6	...	130	mpfi	3.47	2
2	3	1	alfa-romero Quadrifoglio	gas	std	two	hatchback	rwd	front	94.5	...	152	mpfi	2.68	3
3	4	2	audi 100 ls	gas	std	four	sedan	fwd	front	99.8	...	109	mpfi	3.19	3
4	5	2	audi 100ls	gas	std	four	sedan	4wd	front	99.4	...	136	mpfi	3.19	3

rows × 26 columns

Data.tail

```
data.tail()
```

	car_ID	symboling	CarName	fueltype	aspiration	doornumber	carbody	drivewheel	enginelocation	wheelbase	...	enginesize	fuelsystem	boreratio	strok
200	201	-1	volvo 145e (sw)	gas	std	four	sedan	rwd	front	109.1	...	141	mpfi	3.78	3.1
201	202	-1	volvo 144ea	gas	turbo	four	sedan	rwd	front	109.1	...	141	mpfi	3.78	3.1
202	203	-1	volvo 244dl	gas	std	four	sedan	rwd	front	109.1	...	173	mpfi	3.58	2.8
203	204	-1	volvo 246	diesel	turbo	four	sedan	rwd	front	109.1	...	145	idi	3.01	3.4
204	205	-1	volvo 264gl	gas	turbo	four	sedan	rwd	front	109.1	...	141	mpfi	3.78	3.1

Data.shape

```
data.shape
```

```
(205, 26)
```

Data.info

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 205 entries, 0 to 204
Data columns (total 26 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   car_ID           205 non-null    int64
 1   symboling        205 non-null    int64
 2   CarName          205 non-null    object
 3   fueltype         205 non-null    object
 4   aspiration       205 non-null    object
```

```
categorical_cols = ['fueltype','aspiration','doornumber','carbody','drivewheel','enginelocation','enginetype',
    'cylindernumber',
    'fuelsystem'
]

label_encoder = LabelEncoder()
for column in categorical_cols:
    data[column] = label_encoder.fit_transform(data[column])
```

Data.describe

```
data.describe()
```

	car_ID	symboling	wheelbase	carlength	carwidth	carheight	curbweight	enginesize	boreratio	stroke
count	205.000000	205.000000	205.000000	205.000000	205.000000	205.000000	205.000000	205.000000	205.000000	205.000000
mean	103.000000	0.834146	98.756585	174.049268	65.907805	53.724878	2555.565854	126.907317	3.329756	3.255415
std	59.322565	1.245307	6.021776	12.337289	2.145204	2.443522	520.680204	41.642693	0.270844	0.313597
min	1.000000	-2.000000	86.600000	141.100000	60.300000	47.800000	1488.000000	61.000000	2.540000	2.070000
25%	52.000000	0.000000	94.500000	166.300000	64.100000	52.000000	2145.000000	97.000000	3.150000	3.110000
50%	103.000000	1.000000	97.000000	173.200000	65.500000	54.100000	2414.000000	120.000000	3.310000	3.290000
75%	154.000000	2.000000	102.400000	183.100000	66.900000	55.500000	2935.000000	141.000000	3.580000	3.410000
max	205.000000	3.000000	120.900000	208.100000	72.300000	59.800000	4066.000000	326.000000	3.940000	4.170000

Data.columns

```
data.columns
```

```
Index(['car_ID', 'symboling', 'CarName', 'fueltype', 'aspiration',
       'doornumber', 'carbody', 'drivewheel', 'enginelocation', 'wheelbase',
       'carlength', 'carwidth', 'carheight', 'curbweight', 'enginetype',
       'cylindernumber', 'enginesize', 'fuelsystem', 'boreratio', 'stroke',
       'compressionratio', 'horsepower', 'peakrpm', 'citympg', 'highwaympg',
       'price'],
      dtype='object')
```

```
data = data.drop(['car_ID', 'CarName'], axis = 1)
```

```
data.isnull().sum()
```

	0
symboling	0
fueltype	0
aspiration	0
doornumber	0
carbody	0
drivewheel	0
enginelocation	0
wheelbase	0

```
data.drop_duplicates(inplace=True)
data.shape
```

```
(204, 24)
```

Labeling Categorical Columns

```
categorical_cols = ['fueltype','aspiration','doornumber','carbody','drivewheel','enginelocation','enginetype',
    'cylindernumber',
    'fuelsystem'
]

label_encoder = LabelEncoder()
for column in categorical_cols:
    data[column] = label_encoder.fit_transform(data[column])
```

Standardization

```
numerical_cols = ['wheelbase', 'carlength', 'carwidth', 'carheight', 'curbweight',
                  'enginesize', 'boreratio', 'stroke', 'compressionratio', 'horsepower',
                  'peakrpm', 'citympg', 'highwaympg']

scaler = StandardScaler()
data[numerical_cols] = scaler.fit_transform(data[numerical_cols])
```

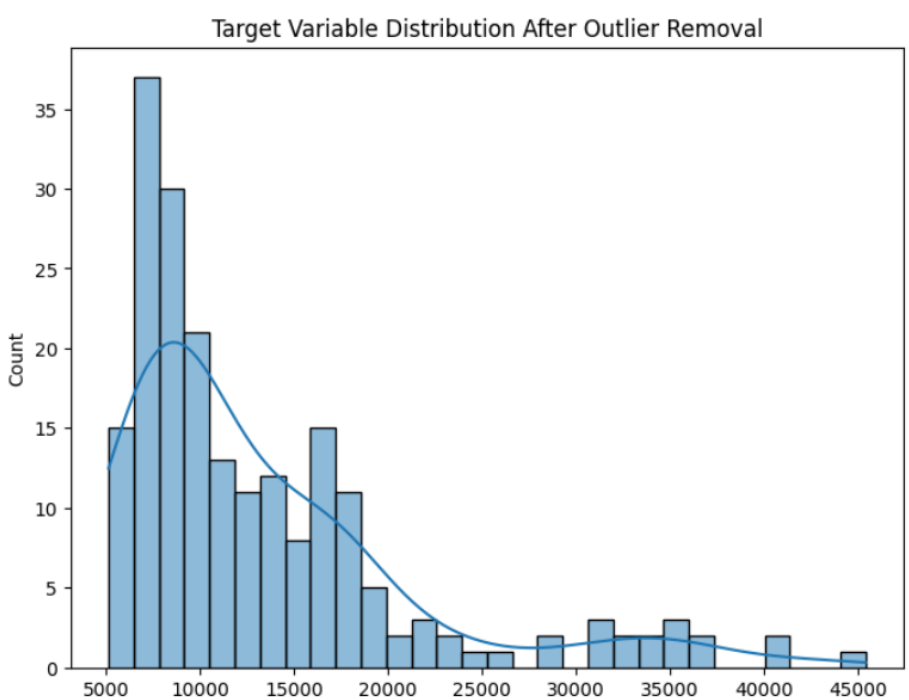
Separating Dependent and Independent Columns

```
x = data.drop(['price'], axis=1)
y = data['price']
```

Displaying Target Values Distribution

```
plt.figure(figsize=(8, 6))
sns.histplot(y, bins=30, kde=True)
plt.title('Target Variable Distribution After Outlier Removal')
plt.xlabel('Energy Consumption (kWh)')
```

```
Text(0.5, 0, 'Energy Consumption (kWh)')
```



Splitting Data

```
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2, random_state=42)
```

Training Model

```
model = LinearRegression()
model.fit(x_train, y_train)
```

Evaluating Accuracy

```
y_pred = model.predict(x_test)
mae = mean_absolute_error(y_test, y_pred)
mse = mean_squared_error(y_test, y_pred)
rmse = np.sqrt(mse)
r2_square = r2_score(y_test,y_pred)

print("\nEvaluation Metrics:")
print(f"MAE:  {mae:.4f}")
print(f"MSE:  {mse:.4f}")
print(f"RMSE: {rmse:.4f}")
print(f"R-squared: {r2_square}")
```

```
Evaluation Metrics:
MAE:  2985.1056
MSE:  18651075.3110
RMSE: 4318.6891
R-squared: 0.7713103044650615
```

Project No 02

Fake Currency Classification

Importing the Dependencies:

```
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
from sklearn.preprocessing import StandardScaler
from imblearn.over_sampling import SMOTE
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score
```

Data Collection and Data Processing:

```
banknote_data = pd.read_csv('/content/drive/MyDrive/Project/data_banknote_authentication.csv')
banknote_data.columns = ['variance', 'asymmetry', 'kurtosis', 'entropy', 'authentication']
```

Data.Head

```
banknote_data.head()
```

	variance	asymmetry	kurtosis	entropy	authentication
0	4.54590	8.1674	-2.4586	-1.46210	0
1	3.86600	-2.6383	1.9242	0.10645	0
2	3.45660	9.5228	-4.0112	-3.59440	0
3	0.32924	-4.4552	4.5718	-0.98880	0
4	4.36840	9.6718	-3.9606	-3.16250	0

Data .Tail

```
banknote_data.tail()
```

	variance	asymmetry	kurtosis	entropy	authentication
1366	0.40614	1.34920	-1.4501	-0.55949	1
1367	-1.38870	-4.87730	6.4774	0.34179	1
1368	-3.75030	-13.45860	17.5932	-2.77710	1
1369	-3.56370	-8.38270	12.3930	-1.28230	1
1370	-2.54190	-0.65804	2.6842	1.19520	1

Data.Shape:

```
banknote_data.shape
```

```
(1347, 5)
```

Data.Describe

```
banknote_data.describe()
```

	variance	asymmetry	kurtosis	entropy	authentication
count	1371.000000	1371.000000	1371.000000	1371.000000	1371.000000
mean	0.431410	1.917434	1.400694	-1.192200	0.444931
std	2.842494	5.868359	4.310105	2.101683	0.497139
min	-7.042100	-13.773100	-5.286100	-8.548200	0.000000
25%	-1.774700	-1.711300	-1.553350	-2.417000	0.000000
50%	0.495710	2.313400	0.616630	-0.586650	0.000000
75%	2.814650	6.813100	3.181600	0.394810	1.000000
max	6.824800	12.951600	17.927400	2.449500	1.000000

Data.Columns

```
banknote_data.columns
```

```
Index(['variance', 'asymmetry', 'kurtosis', 'entropy', 'authentication'], dtype='object')
```

Data Cleaning, Removing Null Values and Removing Duplicates

```
banknote_data.isnull().sum()
```

	0
variance	0
asymmetry	0
kurtosis	0
entropy	0
authentication	0

dtype: int64

```
banknote_data.drop_duplicates(inplace=True)
banknote_data.shape
```

```
(1347, 5)
```

One Hot Encoding

```
numerical_cols = ['variance', 'asymmetry', 'kurtosis', 'entropy']

Q1 = banknote_data[numerical_cols].quantile(0.25)
Q3 = banknote_data[numerical_cols].quantile(0.75)
IQR = Q3 - Q1

outlier_removed_data = banknote_data[~((banknote_data[numerical_cols] < (Q1 - 1.5 * IQR)) | (banknote_data[numerical_cols] > (Q3 + 1.5 * IQR))).any(a

outlier_removed_data.shape
```

Standardization

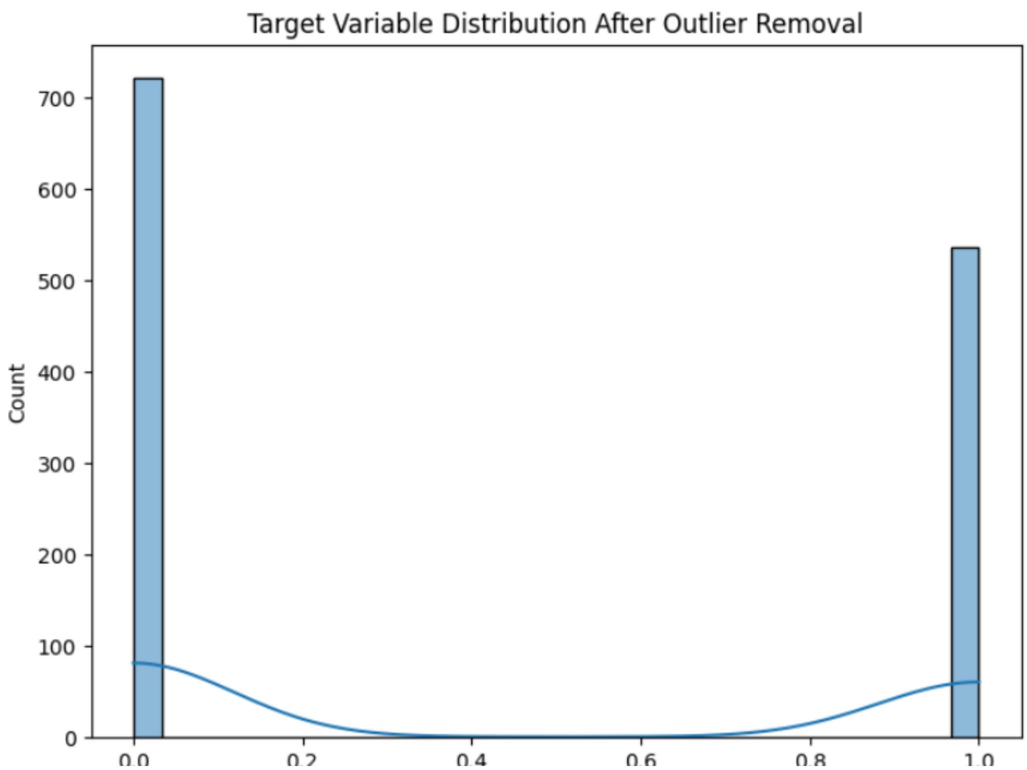
```
X = outlier_removed_data.drop('authentication', axis=1)
Y = outlier_removed_data['authentication']

scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

feature_names = X.columns
X_scaled = pd.DataFrame(X_scaled, columns=feature_names, index=X.index)

X_scaled.head()
```

	variance	asymmetry	kurtosis	entropy
0	1.412546	1.112068	-0.935469	-0.243186
1	1.159769	-0.921273	0.279584	0.580915
2	1.007561	1.367118	-1.365900	-1.363475
3	-0.155142	-1.263165	1.013584	0.005481
4	1.346554	1.395156	-1.351872	-1.136559

Displaying Target Values Distribution

```
plt.figure(figsize=(8, 6))
sns.histplot(Y, bins=30, kde=True)
plt.title('Target Variable Distribution After Outlier Removal')
plt.xlabel('Conterfeit')
```

```
Text(0.5, 0, 'Conterfeit')
```



```
smote = SMOTE(random_state=42)
X_resampled, Y_resampled = smote.fit_resample(X_scaled, Y)
```

```
data_resampled = pd.concat([pd.DataFrame(X_resampled, columns=X.columns), pd.DataFrame(Y_resampled, columns=['authentication'])], axis=1)
data_resampled.head()
```

	variance	asymmetry	kurtosis	entropy	authentication
0	1.412546	1.112068	-0.935469	-0.243186	0
1	1.159769	-0.921273	0.279584	0.580915	0
2	1.007561	1.367118	-1.365900	-1.363475	0
3	-0.155142	-1.263165	1.013584	0.005481	0
4	1.346554	1.395156	-1.351872	-1.136559	0

Splitting Data

```
X = data_resampled.drop("authentication", axis=1)
Y = data_resampled['authentication']
```

```
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size = 0.1, stratify=Y, random_state=42)
```

```
print(X.shape, X_train.shape, X_test.shape)
```

```
(1444, 4) (1299, 4) (145, 4)
```

Training Model

```
model = RandomForestClassifier(random_state=42)
model.fit(X_train, Y_train)
Y_pred = model.predict(X_test)
```

Evaluating Accuracy

```
print("Accuracy:", accuracy_score(Y_test, Y_pred))
print("Classification Report:\n", classification_report(Y_test, Y_pred))
print("Confusion Matrix:\n", confusion_matrix(Y_test, Y_pred))
```

```
Accuracy: 0.9793103448275862
Classification Report:
               precision    recall  f1-score   support

           0       1.00      0.96      0.98        73
           1       0.96      1.00      0.98        72

    accuracy                           0.98       145
   macro avg       0.98      0.98      0.98       145
weighted avg       0.98      0.98      0.98       145

Confusion Matrix:
 [[70  3]
 [ 0 72]]
```