SECTION A

1. The primary use of NumPy library is: Ans.(b) Numerical computation
2. Which algorithm is suitable for binary classification: Ans.(c) Logistic Regression
3. A method to detect outliers is: Ans.(b) Z-score
4. Train-test split is mainly used for: Ans.(b) Avoiding Overfitting
5. An evaluation metric for Regression is: Ans.(d) Mean Squared Error

SECTION B

6. Supervised Learning:Supervised learning uses labeled data, where both input and correct output are known. The model learns by mapping inputs to outputs. Example: Email spam detection (spam vs not spam) Unsupervised Learning:Unsupervised learning works with unlabeled data and finds hidden patterns or structures. Example: Customer segmentation using clustering
7. Feature scaling is the process of bringing all input features to a similar scale so that no single feature dominates due to its large values. Standardization transforms data to have zero mean and unit variance, which helps algorithms like gradient descent and KNN converge faster.
8. Overfitting occurs when a model learns the training data too well, including noise, and performs poorly on unseen data. when the performance gap between training and testing gap is greater than 5% is considered Overfitting and less than 5% is considered Underfitting. Use techniques such as cross-validation, pruning (for trees), dropout, early stopping, and increasing training data.
9. Classification predicts discrete or categorical outputs. The goal is to assign inputs to predefined classes,and also use in categorial data. Regression predicts continuous numerical values,and use for numerical data. The goal is to estimate a quantity.
10. 1.Problem definition and understanding business requirements. 2.Data collection from various sources. 3.Data cleaning and preprocessing. 4.Exploratory Data Analysis (EDA). 5.Feature engineering and selection. 6.Model building and training. 7.Model evaluation and validation. 8.Deployment and monitoring.

SECTION C

Task 1 – Data Preprocessing

In [407…
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

Load dataset using pandas

In [408…
```
ds = pd.read_csv('Used_Bikes.csv')
ds.head()
```

Out[408…

	bike_name	price	city	kms_driven	owner	age	power	brand
0	TVS Star City Plus Dual Tone 110cc	35000.0	Ahmedabad	17654.0	First Owner	3.0	110.0	TVS
1	Royal Enfield Classic 350cc	119900.0	Delhi	11000.0	First Owner	4.0	350.0	Royal Enfield
2	Triumph Daytona 675R	600000.0	Delhi	110.0	First Owner	8.0	675.0	Triumph
3	TVS Apache RTR 180cc	65000.0	Bangalore	16329.0	First Owner	4.0	180.0	TVS
4	Yamaha FZ S V 2.0 150cc-Ltd. Edition	80000.0	Bangalore	10000.0	First Owner	3.0	150.0	Yamaha

Detect and handle missing values

In [409…
```
ds.duplicated().sum()
```

Out[409…
```
np.int64(25324)
```

In [410…
```
ds.drop_duplicates(inplace=True)
```

In [411…
```
ds.duplicated().sum()
```

Out[411…
```
np.int64(0)
```

In [412…
```
ds.isnull()
```

Out[412…

	bike_name	price	city	kms_driven	owner	age	power	brand
0	False	False	False	False	False	False	False	False
1	False	False	False	False	False	False	False	False
2	False	False	False	False	False	False	False	False
3	False	False	False	False	False	False	False	False
4	False	False	False	False	False	False	False	False
...	...	...	...	...	...	...	...	...
9362	False	False	False	False	False	False	False	False
9369	False	False	False	False	False	False	False	False
9370	False	False	False	False	False	False	False	False
9371	False	False	False	False	False	False	False	False
9372	False	False	False	False	False	False	False	False

7324 rows × 8 columns

In [413…
```
ds.isnull().sum()
```

Out[413…
```
bike_name     0
price         0
city          0
kms_driven    0
owner         0
age           0
power         0
brand         0
dtype: int64
```

In [414…
```
# if we have null values we can drop them using dropna() function
ds1 = ds.dropna()
ds1.head()
```

Out[414…

	bike_name	price	city	kms_driven	owner	age	power	brand
0	TVS Star City Plus Dual Tone 110cc	35000.0	Ahmedabad	17654.0	First Owner	3.0	110.0	TVS
1	Royal Enfield Classic 350cc	119900.0	Delhi	11000.0	First Owner	4.0	350.0	Royal Enfield
2	Triumph Daytona 675R	600000.0	Delhi	110.0	First Owner	8.0	675.0	Triumph
3	TVS Apache RTR 180cc	65000.0	Bangalore	16329.0	First Owner	4.0	180.0	TVS
4	Yamaha FZ S V 2.0 150cc-Ltd. Edition	80000.0	Bangalore	10000.0	First Owner	3.0	150.0	Yamaha

In [415…
```
# now we fill the null values
ds.fillna(ds.mean(numeric_only=True))
ds.head()
```

Out[415…

	bike_name	price	city	kms_driven	owner	age	power	brand
0	TVS Star City Plus Dual Tone 110cc	35000.0	Ahmedabad	17654.0	First Owner	3.0	110.0	TVS
1	Royal Enfield Classic 350cc	119900.0	Delhi	11000.0	First Owner	4.0	350.0	Royal Enfield
2	Triumph Daytona 675R	600000.0	Delhi	110.0	First Owner	8.0	675.0	Triumph
3	TVS Apache RTR 180cc	65000.0	Bangalore	16329.0	First Owner	4.0	180.0	TVS
4	Yamaha FZ S V 2.0 150cc-Ltd. Edition	80000.0	Bangalore	10000.0	First Owner	3.0	150.0	Yamaha

Summary Insights : The dataset was successfully loaded using the pandas library, allowing structured data manipulation and analysis. Missing values were identified using null-value checks and handled appropriately by either removing incomplete records or imputing values using statistical measures such as mean or median. Numerical features were analyzed for outliers to ensure data quality and reliability. Handling missing values and outliers improved data consistency and prepared the dataset for accurate and stable machine learning modeling. The Z-score method was used to identify values far from the mean, which is effective when data follows a normal distribution.

Task 2 – Model Building

Train-test split

In [416…
```
x = pd.get_dummies(x, drop_first=True)
```

In [417…
```
ds.fillna(ds.mean(numeric_only=True), inplace=True)
```

In [418…
```
dt = {"First Owner":1,"Second Owner":2,"Third Owner":3,"Fourth Owner Or More":4}
dt
```

Out[418…
```
{'First Owner': 1,
 'Second Owner': 2,
 'Third Owner': 3,
 'Fourth Owner Or More': 4}
```

In [419…
```
ds['owner']=ds['owner'].map(dt)
```

In [420…
```
ds.head()
```

Out[420…

	bike_name	price	city	kms_driven	owner	age	power	brand
0	TVS Star City Plus Dual Tone 110cc	35000.0	Ahmedabad	17654.0	1	3.0	110.0	TVS
1	Royal Enfield Classic 350cc	119900.0	Delhi	11000.0	1	4.0	350.0	Royal Enfield
2	Triumph Daytona 675R	600000.0	Delhi	110.0	1	8.0	675.0	Triumph
3	TVS Apache RTR 180cc	65000.0	Bangalore	16329.0	1	4.0	180.0	TVS
4	Yamaha FZ S V 2.0 150cc-Ltd. Edition	80000.0	Bangalore	10000.0	1	3.0	150.0	Yamaha

In [421…
```
brands=["TVS","Yamaha","Bajaj","Hero","Royal Enfield","Honda","Suzuki","TVS","KTM","Harley Davidson","Kawasaki","Mahindra","Hyosung","Benelli","Triumph","Ducati","BMW"]
ds[ds['brand'].isin(brands)]
```

Out[421…

	bike_name	price	city	kms_driven	owner	age	power	brand
0	TVS Star City Plus Dual Tone 110cc	35000.0	Ahmedabad	17654.0	1	3.0	110.0	TVS
1	Royal Enfield Classic 350cc	119900.0	Delhi	11000.0	1	4.0	350.0	Royal Enfield
2	Triumph Daytona 675R	600000.0	Delhi	110.0	1	8.0	675.0	Triumph
3	TVS Apache RTR 180cc	65000.0	Bangalore	16329.0	1	4.0	180.0	TVS
4	Yamaha FZ S V 2.0 150cc-Ltd. Edition	80000.0	Bangalore	10000.0	1	3.0	150.0	Yamaha
...	...	...	...	...	...	...	...	...
9361	Bajaj Avenger 220cc	50000.0	Bangalore	29134.0	1	7.0	220.0	Bajaj
9362	Hero Hunk Rear Disc 150cc	25000.0	Delhi	48587.0	1	8.0	150.0	Hero
9369	Bajaj Avenger 220cc	35000.0	Bangalore	60000.0	1	9.0	220.0	Bajaj
9371	Bajaj Dominar 400 ABS	139000.0	Hyderabad	21300.0	1	4.0	400.0	Bajaj
9372	Bajaj Avenger Street 220	80000.0	Hyderabad	7127.0	1	5.0	220.0	Bajaj

7216 rows × 8 columns

In [422…
```
brand_ls = list(ds['brand'].value_counts().head(16).keys())
brand_ls
```

Out[422…
```
['Bajaj',
 'Royal Enfield',
 'Hero',
 'Honda',
 'Yamaha',
 'TVS',
 'KTM',
 'Suzuki',
 'Harley-Davidson',
 'Kawasaki',
 'Hyosung',
 'Mahindra',
 'Benelli',
 'Triumph',
 'Ducati',
 'BMW']
```

In [423…
```
ds2 = ds[ds['brand'].isin(brand_ls)]
ds2.head()
```

Out[423…

	bike_name	price	city	kms_driven	owner	age	power	brand
0	TVS Star City Plus Dual Tone 110cc	35000.0	Ahmedabad	17654.0	1	3.0	110.0	TVS
1	Royal Enfield Classic 350cc	119900.0	Delhi	11000.0	1	4.0	350.0	Royal Enfield
2	Triumph Daytona 675R	600000.0	Delhi	110.0	1	8.0	675.0	Triumph
3	TVS Apache RTR 180cc	65000.0	Bangalore	16329.0	1	4.0	180.0	TVS
4	Yamaha FZ S V 2.0 150cc-Ltd. Edition	80000.0	Bangalore	10000.0	1	3.0	150.0	Yamaha

In [424…
```
ds2['brand'].value_counts()
```

Out[424…
```
brand
Bajaj              2081
Royal Enfield      1346
Hero               1142
Honda               676
Yamaha              651
TVS                 481
KTM                 375
Suzuki              203
Harley-Davidson      91
Kawasaki             61
Hyosung              53
Mahindra             50
Benelli              46
Triumph              21
Ducati               20
BMW                  10
Name: count, dtype: int64
```

In [425…
```
brand_dict = {brand:i for i, brand in enumerate(list(ds2['brand'].value_counts().head(16).keys()),start = 1)}
print(brand_dict)
ds2['brand']=ds2['brand'].map(brand_dict)
```

```
{'Bajaj': 1, 'Royal Enfield': 2, 'Hero': 3, 'Honda': 4, 'Yamaha': 5, 'TVS': 6, 'KTM': 7, 'Suzuki': 8, 'Harley-Davidson': 9, 'Kawasaki': 10, 'Hyosung': 11, 'Mahindra': 12, 'Benelli': 13, 'Triumph': 14, 'Ducati': 15, 'BMW': 16}
/var/folders/lc/vzgpk1x90wj13660t2sjm_w80000gn/T/ipykernel_34389/844270164.py:3: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  ds2['brand']=ds2['brand'].map(brand_dict)
```

In [426…
```
ds2.head()
```

Out[426…

	bike_name	price	city	kms_driven	owner	age	power	brand
0	TVS Star City Plus Dual Tone 110cc	35000.0	Ahmedabad	17654.0	1	3.0	110.0	6
1	Royal Enfield Classic 350cc	119900.0	Delhi	11000.0	1	4.0	350.0	2
2	Triumph Daytona 675R	600000.0	Delhi	110.0	1	8.0	675.0	14
3	TVS Apache RTR 180cc	65000.0	Bangalore	16329.0	1	4.0	180.0	6
4	Yamaha FZ S V 2.0 150cc-Ltd. Edition	80000.0	Bangalore	10000.0	1	3.0	150.0	5

In [427…
```
ds2.drop(['bike_name','city'],axis=1,inplace=True)
```

```
/var/folders/lc/vzgpk1x90wj13660t2sjm_w80000gn/T/ipykernel_34389/2965310615.py:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  ds2.drop(['bike_name','city'],axis=1,inplace=True)
```

In [428…
```
ds2.head()
```

Out[428…

	price	kms_driven	owner	age	power	brand
0	35000.0	17654.0	1	3.0	110.0	6
1	119900.0	11000.0	1	4.0	350.0	2
2	600000.0	110.0	1	8.0	675.0	14
3	65000.0	16329.0	1	4.0	180.0	6
4	80000.0	10000.0	1	3.0	150.0	5

In [429…
```
x = ds2.drop('price',axis=1)
y = ds2[['price']]
```

In [430…
```
y
```

Out[430…

	price
0	35000.0
1	119900.0
2	600000.0
3	65000.0
4	80000.0
...	...
9362	25000.0
9369	35000.0
9370	450000.0
9371	139000.0
9372	80000.0

7307 rows × 1 columns

In [431…
```
x
```

Out[431…

	kms_driven	owner	age	power	brand
0	17654.0	1	3.0	110.0	6
1	11000.0	1	4.0	350.0	2
2	110.0	1	8.0	675.0	14
3	16329.0	1	4.0	180.0	6
4	10000.0	1	3.0	150.0	5
...	...	...	...	...	...
9362	48587.0	1	8.0	150.0	3
9369	60000.0	1	9.0	220.0	1
9370	3430.0	1	4.0	750.0	9
9371	21300.0	1	4.0	400.0	1
9372	7127.0	1	5.0	220.0	1

7307 rows × 5 columns

In [432…
```
from sklearn.model_selection import train_test_split
```

In [433…
```
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.2)
```

In [434…
```
x_train
```

Out[434…

	kms_driven	owner	age	power	brand
1126	1300.0	1	1.0	200.0	6
5281	38950.0	1	5.0	110.0	6
1670	44000.0	1	8.0	350.0	2
4265	16270.0	1	5.0	100.0	3
1155	41736.0	1	7.0	220.0	1
...	...	...	...	...	...
2970	5095.0	1	5.0	150.0	1
2170	15000.0	1	5.0	350.0	2
1665	29299.0	1	4.0	350.0	2
7822	28350.0	1	15.0	150.0	6
8480	19500.0	1	5.0	500.0	2

5845 rows × 5 columns

In [435…
```
x_test
```

Out[435…

	kms_driven	owner	age	power	brand
2905	25000.0	1	9.0	100.0	3
7178	30000.0	1	5.0	160.0	4
5831	18692.0	1	3.0	150.0	5
4027	29508.0	1	8.0	125.0	4
1278	6600.0	1	18.0	350.0	2
...	...	...	...	...	...
7666	27629.0	1	10.0	150.0	5
5469	24000.0	1	7.0	350.0	2
9029	50000.0	2	19.0	100.0	3
4147	24200.0	1	4.0	400.0	1
3886	21000.0	1	4.0	220.0	1

1462 rows × 5 columns

In [436…
```
y_train
```

Out[436…

	price
1126	130000.0
5281	40000.0
1670	81000.0
4265	33000.0
1155	38000.0
...	...
2970	41000.0
2170	120000.0
1665	111500.0
7822	16000.0
8480	125000.0

5845 rows × 1 columns

In [437…
```
y_test
```

Out[437…

	price
2905	27500.0
7178	70000.0
5831	110000.0
4027	30000.0
1278	130000.0
...	...
7666	33000.0
5469	75000.0
9029	14000.0
4147	120000.0
3886	61700.0

1462 rows × 1 columns

Train Linear Regression model

In [438…
```
from sklearn.linear_model import LinearRegression
```

In [439…
```
lnr = LinearRegression()
```

In [440…
```
lnr.fit(x_train, y_train)
```

Out[440…

▾ LinearRegression

▶ Parameters

Predictions & Evaluation

In [441…
```
from sklearn.metrics import mean_squared_error, r2_score
```

In [442…
```
y_pred = lnr.predict(x_test)
```

In [443…
```
print("Mean Squared Error (MSE):", mean_squared_error(y_test, y_pred))
print("R² Score:", r2_score(y_test, y_pred))
```

```
Mean Squared Error (MSE): 3779659357.4343724
R² Score: 0.7291625439585994
```

Interpretation : • The dataset was divided into training and testing sets to evaluate model generalization. • Categorical features were converted into numerical form using one-hot encoding. • A Linear Regression model was trained to predict used bike prices. • Model performance was assessed using Mean Squared Error and R^2 score. • Lower MSE indicates reduced prediction error. • A higher R^2 score shows that the model explains a large portion of price variability. • Proper preprocessing contributed to stable and reliable predictions.