MARMARA UNIVERSITY
FACULTY OF ENGINEERING

CSE4074.1 - Machine Learning

Term Project – Data Pre-processing Report

Due Date: 06.12.2024

	Name Surname	Student Id
1	Ahmet Arda Nalbant	150121004
2	Umut Bayar	150120043
3	Niyazi Ozan Ateş	150121991
4	Murat Albayrak	150120025
5	Kadir Pekdemir	150121069

1) Data Informations

First, we imported the necessary libraries for data processing and analysis.

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.preprocessing import LabelEncoder
import seaborn as sns
from sklearn.ensemble import RandomForestClassifier
```

Next, we assigned our dataset, stored in CSV format, to a variable named “df.” To facilitate the exploration of the dataset, we displayed the first five rows to quickly inspect its structure.

```
df=pd.read_csv("car_prices.csv")
```

```
df.head()
```

Dataset Display:

	year	make	model	trim	body	transmission	vin	state	condition	odometer	color	interior	seller	mmr	sellingprice	saledate
0	2015	Kia	Sorento	LX	SUV	automatic	5xyktca69fg566472	ca	5.0	16639.0	white	black	kia motors america inc	20500.0	21500.0	Tue Dec 16 2014 12:30:00 GMT-0800 (PST)
1	2015	Kia	Sorento	LX	SUV	automatic	5xyktca69fg561319	ca	5.0	9393.0	white	beige	kia motors america inc	20800.0	21500.0	Tue Dec 16 2014 12:30:00 GMT-0800 (PST)
2	2014	BMW	3 Series	328i SULEV	Sedan	automatic	wba3c1c51ek116351	ca	45.0	1331.0	gray	black	financial services remarketing (lease)	31900.0	30000.0	Thu Jan 15 2015 04:30:00 GMT-0800 (PST)
3	2015	Volvo	S60	T5	Sedan	automatic	yv1612tb4f1310987	ca	41.0	14282.0	white	black	volvo na rep/world omni	27500.0	27750.0	Thu Jan 29 2015 04:30:00 GMT-0800 (PST)
4	2014	BMW	6 Series Gran Coupe	650i	Sedan	automatic	wba6b2c57ed129731	ca	43.0	2641.0	gray	black	financial services remarketing (lease)	66000.0	67000.0	Thu Dec 18 2014 12:30:00 GMT-0800 (PST)

Figure 1

2) Column Descriptions

1. **year**: The year of manufacture of vehicles.
2. **make**: The manufacturer or brand of vehicle (e.g., Kia).
3. **model**: The model name of the vehicle (e.g., Sorento).
4. **trim**: The specific version or configuration of the vehicle model (e.g., LX).
5. **body**: The body type of vehicle (e.g., SUV).
6. **transmission**: The type of transmission system (e.g., automatic or manual).
7. **vin**: The unique Vehicle Identification Number for each vehicle.
8. **state**: The location (state) where the vehicle was sold or listed (e.g., "ca" for California).
9. **condition**: The condition rating of the vehicle, typically on a scale of 1-5 (1: poor, 5: excellent).
10. **odometer**: The mileage recorded on the vehicle's odometer, representing how many miles it has traveled.
11. **color**: The exterior color of the vehicle (e.g., white).
12. **interior**: The interior color of the vehicle (e.g., black or beige).
13. **seller**: The name of the seller or dealership (e.g., Kia Motors America Inc).
14. **mmr**: The Manheim Market Report value, a wholesale price benchmark for the vehicle.
15. **sellingprice**: The actual selling price of the vehicle.
16. **saledate**: The date and time when the vehicle was sold

Following this, we used the df.info() command to display an overview of the dataset's features.

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 558837 entries, 0 to 558836
Data columns (total 16 columns):
 #   Column        Non-Null Count   Dtype  
---  ------        --------------   -----  
 0   year          558837 non-null  int64  
 1   make          548536 non-null  object 
 2   model         548438 non-null  object 
 3   trim          548186 non-null  object 
 4   body          545642 non-null  object 
 5   transmission  493485 non-null  object 
 6   vin           558833 non-null  object 
 7   state         558837 non-null  object 
 8   condition     547017 non-null  float64
 9   odometer      558743 non-null  float64
 10  color         558088 non-null  object 
 11  interior      558088 non-null  object 
 12  seller        558837 non-null  object 
 13  mmr           558799 non-null  float64
 14  sellingprice  558825 non-null  float64
 15  saledate      558825 non-null  object 
dtypes: float64(4), int64(1), object(11)
```

Figure 2

3) Data Preprocessing

3.1) Dropping columns that will not be useful in model training

The **vin** (Vehicle Identification Number) and **saledate** columns were dropped, as they do not contribute directly to price prediction.

```
df.drop(columns=['vin','saledate'], inplace=True)
```

3.2) Replacing null values

Using the command df.isnull().sum(), we identified the columns with missing values.

```
df.isnull().sum()
```

```
year              0
make          10301
model         10399
trim          10651
body          13195
transmission  65352
vin               4
state             0
condition     11820
odometer         94
color           749
interior        749
seller            0
mmr              38
sellingprice     12
saledate         12
dtype: int64
```

Figure 3

To handle these missing values, we filled the numeric columns (year, condition, odometer, mmr, sellingprice) with the mean of each respective column. As a result, the count of null values in the numeric columns was corrected.

```
df['year'].fillna(df['year'].mean())
df['condition'].fillna(df['condition'].mean())
df['odometer'].fillna(df['odometer'].mean())
df['mmr'].fillna(df['mmr'].mean())
df['sellingprice'].fillna(df['sellingprice'].mean())
```

Now, the count of null values in the numeric columns has been corrected.

```
year              0
make          10301
model         10399
trim          10651
body          13195
transmission  65352
vin               4
state             0
condition         0
odometer          0
color           749
interior        749
seller            0
mmr               0
sellingprice      0
saledate         12
dtype: int64
```

Figure 4

For columns with object data types, we replaced missing values with the placeholder value "unknown."

```
object_columns = df.select_dtypes(include=['object']).columns
df[object_columns] = df[object_columns].fillna('Unknown')
```

3.3) Convert categorical variables to numeric values

We used Target encoding to convert the categorical columns (those with object data types) into numeric values, allowing them to be used in model training.

```
categorical_columns = ['make', 'model', 'trim', 'body', 'transmission', 'state', 'color', 'interior', 'seller']
target_column = "sellingprice"
for col in categorical_columns:

    target_encoding = df.groupby(col)[target_column].mean().to_dict()
    df[f"{col}_encoded"] = df[col].map(target_encoding)

df.drop(categorical_columns, axis=1, inplace=True)
```

```
df.head()
```

	year	condition	odometer	mmr	sellingprice	make_encoded	model_encoded	trim_encoded	body_encoded	transmission_enco
0	2015	5.0	16639.0	20500.0	21500.0	11808.872364	14643.257923	10289.386104	15905.503680	13540.408
1	2015	5.0	9393.0	20800.0	21500.0	11808.872364	14643.257923	10289.386104	15905.503680	13540.408
2	2014	45.0	1331.0	31900.0	30000.0	21441.895748	16809.633106	29540.000000	11594.050401	13540.408
3	2015	41.0	14282.0	27500.0	27750.0	11463.952482	13766.587164	16348.906742	11594.050401	13540.408
4	2014	43.0	2641.0	66000.0	67000.0	21441.895748	55162.990741	36178.477223	11594.050401	13540.408

4) Data Visualization

The purpose of creating the following visualizations is to better understand the data, identify patterns, and prepare for model training by exploring relationships between variables and detecting any potential outliers or trends.

Histogram

A histogram is used to visualize the distribution of a single variable, in this case, the **sellingprice** of the vehicles. It groups data into bins and displays the frequency of values within each bin. This helps us understand how the selling prices are distributed (e.g., are they skewed, concentrated in certain ranges, or spread out?). By examining the histogram, we can identify whether the prices are normally distributed, whether there are any pricing clusters, and if any preprocessing like normalization might be required.

```
plt.hist(df['sellingprice'], bins=30, edgecolor='black')
plt.title('Selling Price Distribution')
plt.xlabel('Selling Price')
plt.ylabel('Frequency')
plt.show()
```

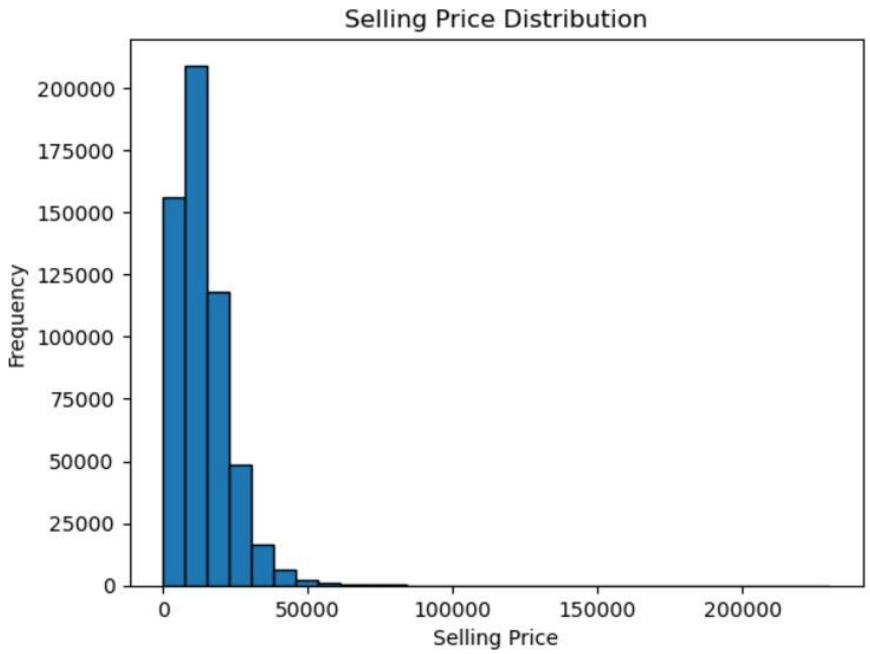

Figure 5

Scatter Plot

A scatter plot is effective for visualizing the relationship between two continuous variables. Here, we plotted **odometer** (mileage) against **sellingprice**. The scatter plot helps us

understand whether there's a linear or non-linear relationship between these two variables, which is important for feature selection in predictive modeling. For example, if there's a clear trend showing that vehicles with lower mileage tend to have higher selling prices, we may want to emphasize this feature in the model.

To explore the relationship between two continuous variables, we created a scatter plot showing the correlation between **odometer** and **sellingprice**.

```
plt.scatter(df['odometer'], df['sellingprice'], alpha=0.5)
plt.title('Selling Price vs Odometer')
plt.xlabel('Odometer')
plt.ylabel('Selling Price')
plt.show()
```

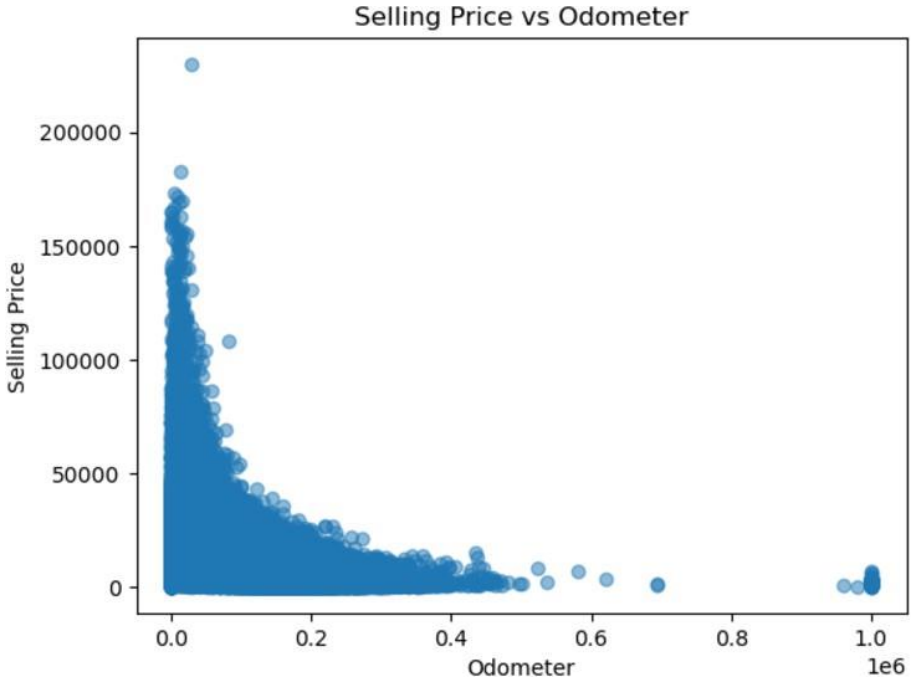

Figure 6

Correlation Matrix

A correlation matrix is used to examine the relationships between multiple numerical variables simultaneously. It provides a heatmap of correlation coefficients, which range from -1 (perfect negative correlation) to 1 (perfect positive correlation), with 0 meaning no correlation. This is crucial for understanding which variables are strongly correlated with each other. For example, if **year** and **sellingprice** are highly positively correlated, it may indicate

that newer cars tend to have higher selling prices. Identifying such correlations helps in feature selection and can inform decisions about which variables to include or exclude in the model.

```
correlation_matrix = df.corr()

sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt='.2f', annot_kws={"size": 7})
plt.title('Correlation Matrix')
plt.show()
```

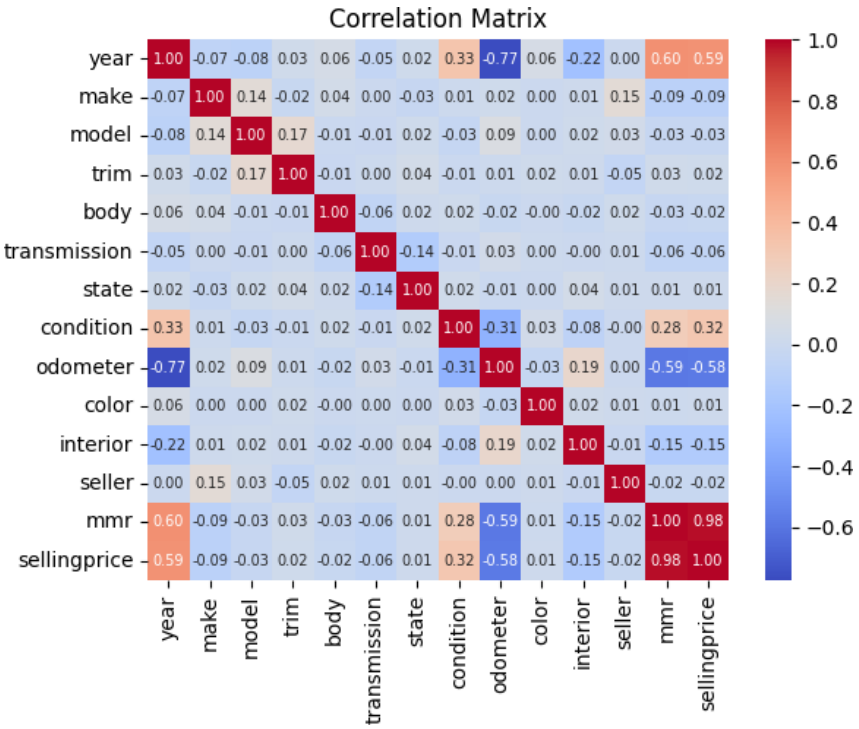

Figure 7

Box Plot

A box plot is a powerful visualization for understanding the distribution of data, especially when we are interested in detecting outliers and central tendencies (e.g., median and quartiles). It displays the minimum, first quartile (25th percentile), median (50th percentile), third quartile (75th percentile), and maximum values. Outliers are typically visualized as points outside the "whiskers" of the plot. For example, a box plot for **odometer** can show whether there are any extreme values or outliers, helping us decide whether to transform or remove these outliers before training a model.

```
plt.figure(figsize=(10, 6))
sns.boxplot(data=df, x='odometer', color='skyblue')

# Adding labels and title
plt.title('Box Plot of Odometer', fontsize=16)
plt.xlabel('Odometer', fontsize=14)
plt.grid(axis='x', linestyle='--', alpha=0.7)

plt.tight_layout()
plt.show()
```

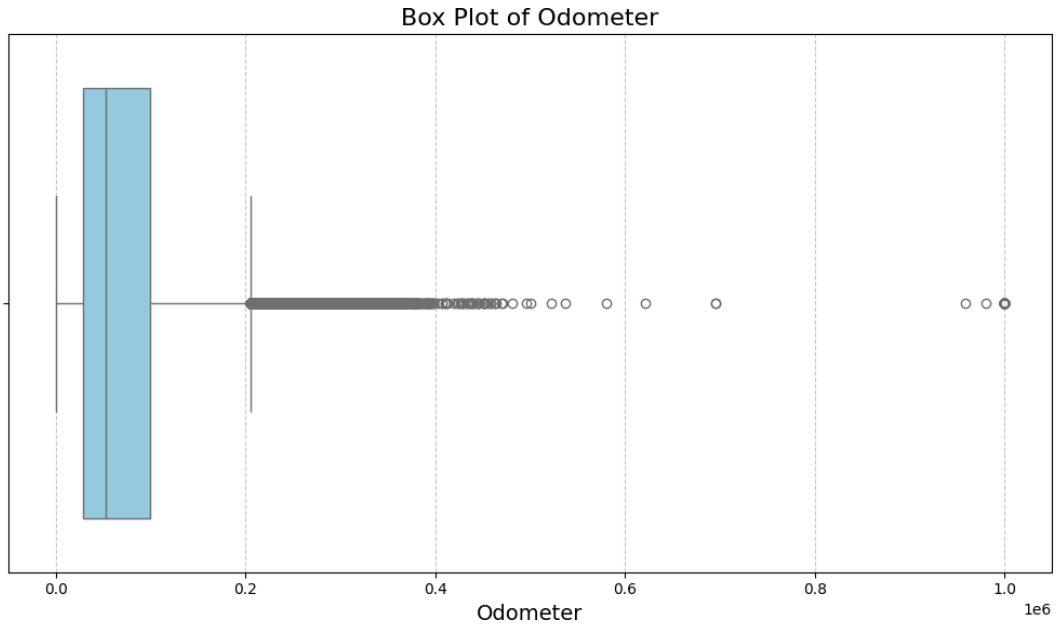

Figure 8