1) Data Informations

First, we describe the necessary libraries.

```
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
```

After this step, we assigned our dataset, which is in CSV format, to a variable named "df". To easily examine the columns in the dataset, we displayed the first 5 rows.

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

```
df.head()
```

Display of our dataset;

	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

Then, we showed the features of our data set with the df.info() command.

```
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

“vın” is the Vehicle identification number. It does not directly contribute to price prediction

Also “saledate” not directly contribute to price prediction.

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

3.2) Replacing null values

Using the command df.isnull().sum(), we identified the number of null values present in each column of the dataset.

```
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

After this step, we try to fill numeric null values with mean values. Our numeric columns are “year”, “condition”, “odometer”, “mmr”, “sellingprice”.

```
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

We replaced the null values in columns with object data type with “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 converted the columns of the object data type into numerical values. We used target encoding for this. The reason we use target encoding in this case is due to having a large number of categorical variables. By doing so, the same variables within the same column are evaluated based on the average of their corresponding selling price values

```
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)
```

4 Data Visulization

The histogram shows how many groups the data is collected in and how the distribution is. Here we created a histogram chart for selling price.

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

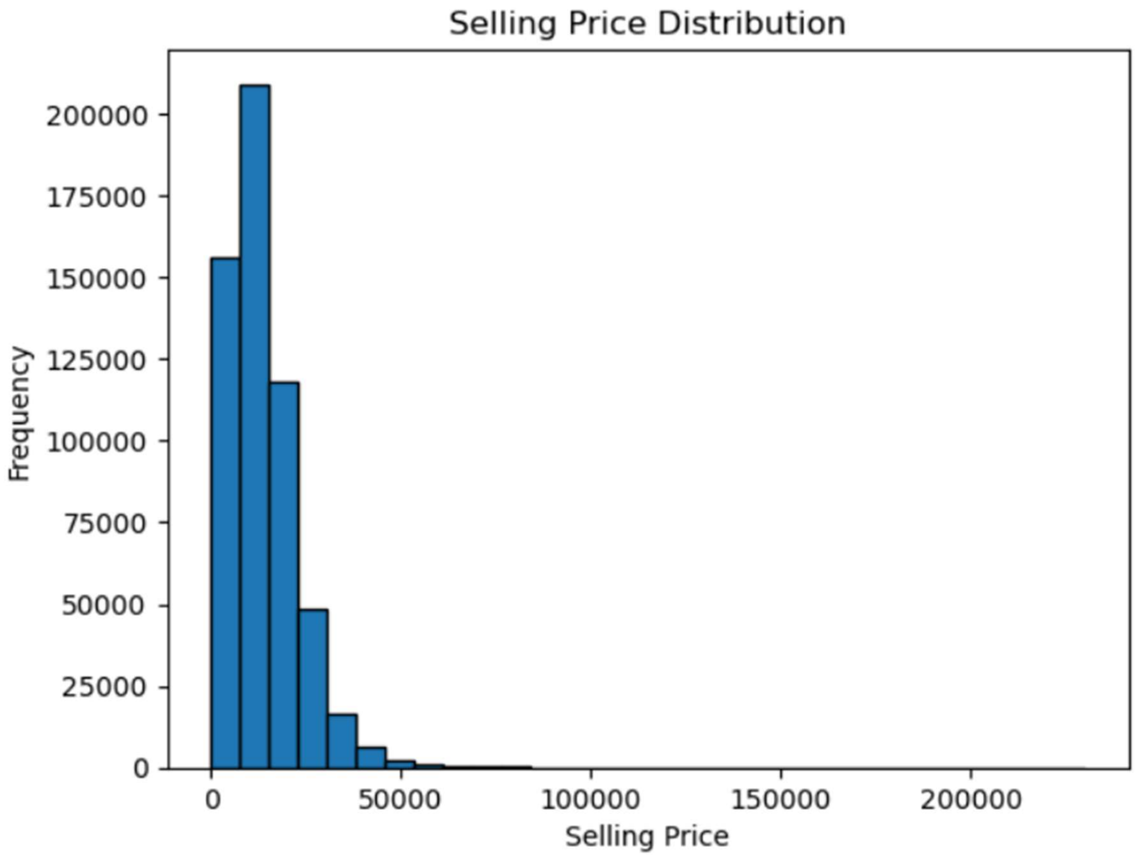

Figure 5

Scatter plot serves to visualize the relationship between two continuous variables. Here we created a scatter plot chart showing the relationship between autometer and selling price.

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

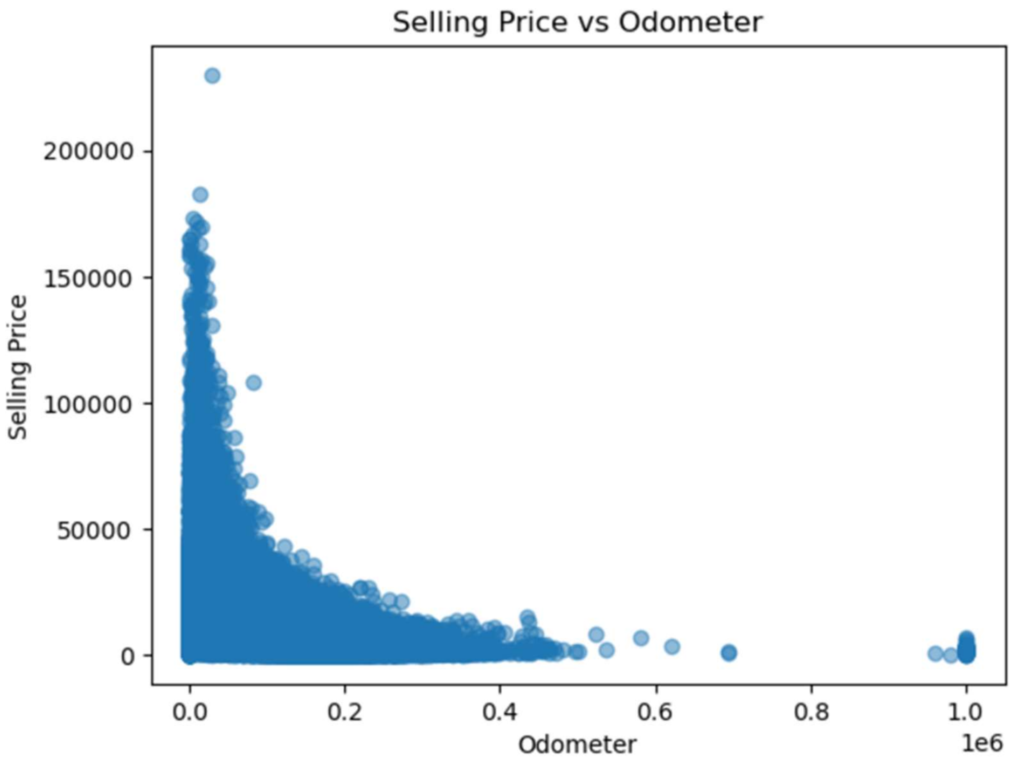

Figure 6

The correlation matrix is used to examine the relationship between numerical variables. Correlation coefficient shows the linear relationship between two variables

```
correlation_matrix = df.corr()

sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt='.2f')
plt.title('Correlation Matrix')
plt.show()
```

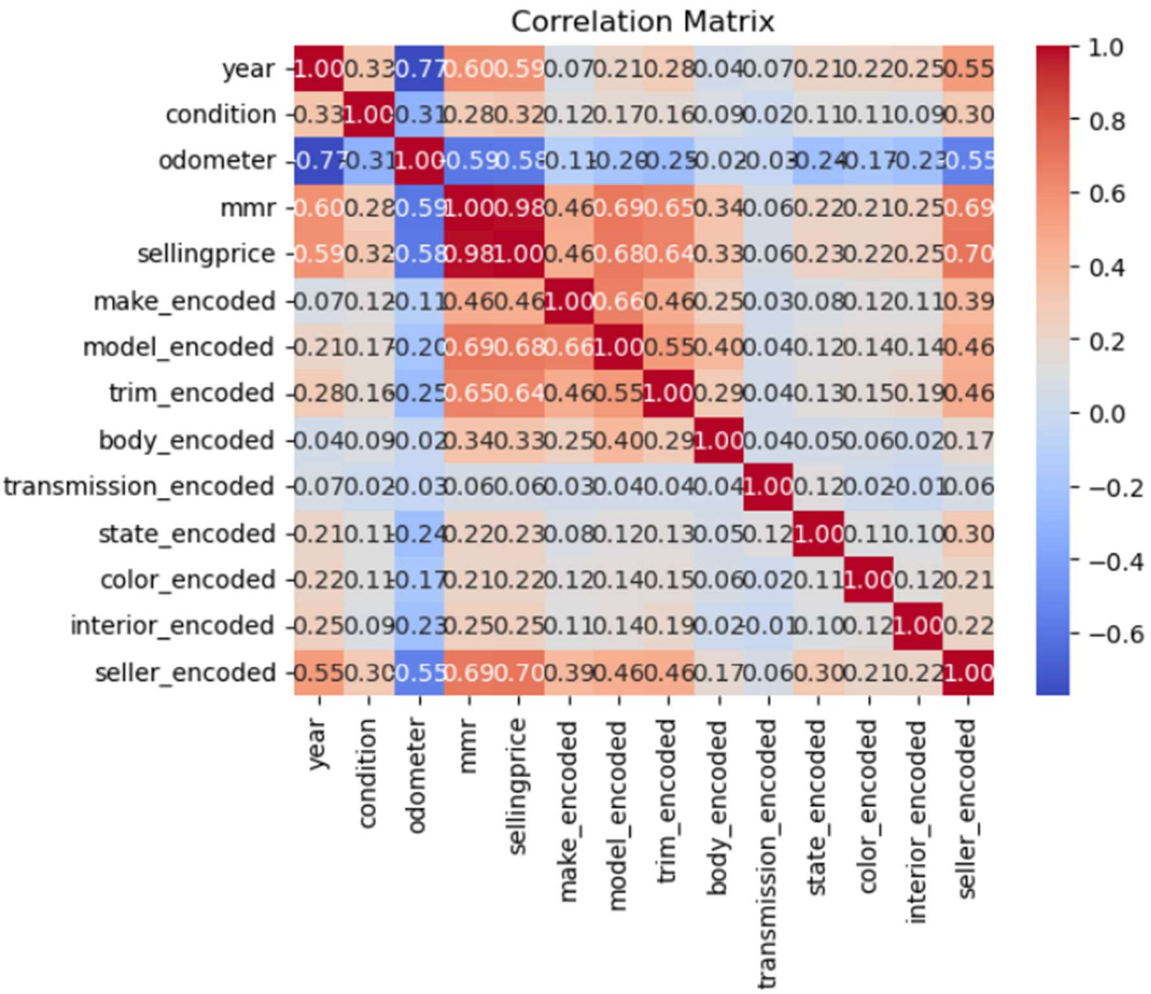

Figure 7

A box plot visualizes the central tendencies (median, quartiles) and outliers of a variable.

```
sns.boxplot(x=df['sellingprice'])
plt.title('Selling Price Boxplot')
plt.show()
```

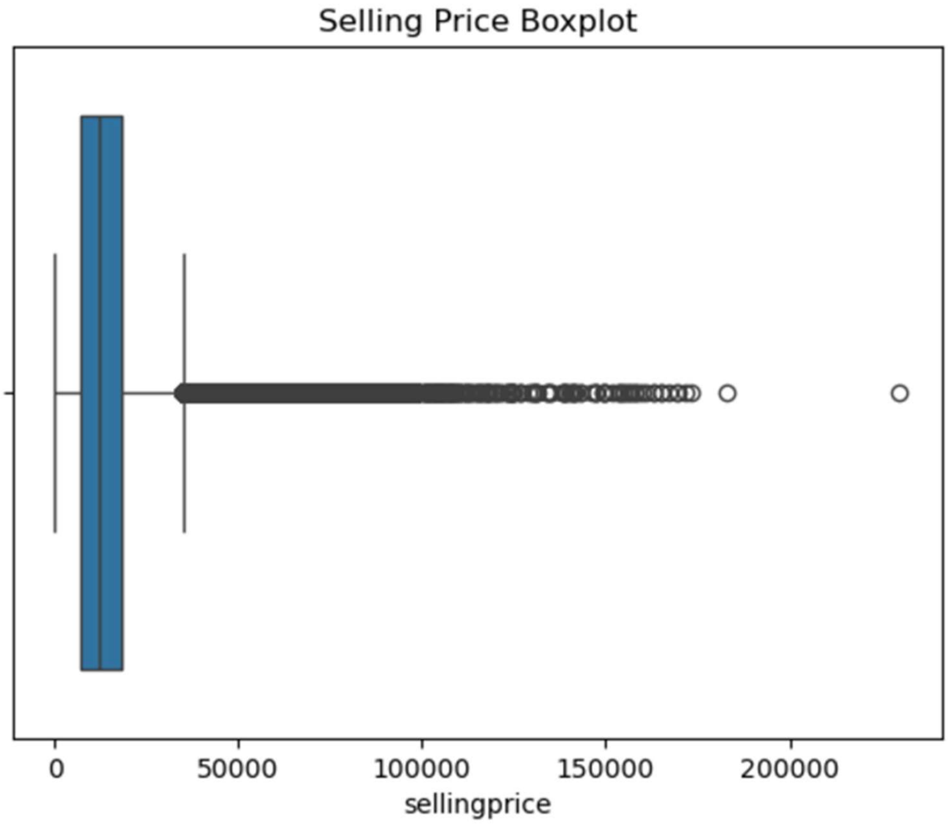
Selling Price Boxplot
0
50000
100000
150000
200000
sellingprice

Figure 8