In [248…

```
import pandas as pd
import numpy as num
import matplotlib.pyplot as mat
import seaborn as sea
from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import MinMaxScaler
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeRegressor
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score, precision_score, recall_score,f1_score
```

In [249…

```
df = pd.read_csv('C:/Users/mohieldine/Desktop/Python Tutorial/Indian Used Cars/usedCar
df.head(100)
```

Out[249]:

	Id	Company	Model	Variant	FuelType	Colour	Kilometer	BodyStyle	Tra
0	555675	MARUTI SUZUKI	CELERIO(2017-2019)	1.0 ZXI AMT O	PETROL	Silver	33197	HATCHBACK	
1	556383	MARUTI SUZUKI	ALTO	LXI	PETROL	Red	10322	HATCHBACK	
2	556422	HYUNDAI	GRAND I10	1.2 KAPPA ASTA	PETROL	Grey	37889	HATCHBACK	
3	556771	TATA	NEXON	XT PLUS	PETROL	A Blue	13106	HATCHBACK	
4	559619	FORD	FIGO	EXI DURATORQ 1.4	DIESEL	Silver	104614	HATCHBACK	
...	...	...	...	...	...	...	...	...	
95	585018	VOLKSWAGEN	POLO	GT TSI 1.2 PETROL AT	PETROL	White	63088	HATCHBACK	
96	585031	HYUNDAI	ELITE I20	SPORTZ 1.2	PETROL	Red	45607	HATCHBACK	
97	585039	FORD	ECOSPORT(2017-2019)	TREND 1.5L TDCI	DIESEL	Brown	63212	SUV	
98	585066	NISSAN	KICKS	XV 1.5	PETROL	White	40924	SUV	
99	585068	HYUNDAI	SANTRO	1.1 LS	PETROL	Silver	86140	HATCHBACK	

100 rows × 19 columns

In [250…

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

```
In [251…  df.drop(columns=['Variant'], inplace=True)
```

```
In [252…  df['FuelType'] = df['FuelType'].replace({'manual': 'Manual'})
```

```
In [253…  df.shape

Out[253]: (1064, 17)
```

```
In [254…  df.columns

Out[254]: Index(['Company', 'Model', 'FuelType', 'Colour', 'Kilometer', 'BodyStyle',
                 'TransmissionType', 'ManufactureDate', 'ModelYear', 'CngKit', 'Price',
                 'Owner', 'DealerState', 'DealerName', 'City', 'Warranty',
                 'QualityScore'],
                dtype='object')
```

```
In [255…  df.duplicated().sum()

Out[255]: 0
```

```
In [256…  df.isnull().sum()

Out[256]: Company                0
          Model                  0
          FuelType               1
          Colour                 0
          Kilometer              0
          BodyStyle              0
          TransmissionType     714
          ManufactureDate        0
          ModelYear              0
          CngKit              1042
          Price                  0
          Owner                  0
          DealerState            0
          DealerName             0
          City                   0
          Warranty               0
          QualityScore           0
          dtype: int64
```

```
In [257…  df.drop(columns='TransmissionType',inplace = True)
          df.drop(columns='CngKit',inplace = True)
```

```
In [258…  df.shape

Out[258]: (1064, 15)
```

```
In [259…  df.head(20)
```

Out[259]:

	Company	Model	FuelType	Colour	Kilometer	BodyStyle	ManufactureDate	Moc
0	MARUTI SUZUKI	CELERIO(2017-2019)	PETROL	Silver	33197	HATCHBACK	2018-02-01	
1	MARUTI SUZUKI	ALTO	PETROL	Red	10322	HATCHBACK	2021-03-01	
2	HYUNDAI	GRAND I10	PETROL	Grey	37889	HATCHBACK	2015-03-01	
3	TATA	NEXON	PETROL	A Blue	13106	HATCHBACK	2020-08-01	
4	FORD	FIGO	DIESEL	Silver	104614	HATCHBACK	2010-11-01	
5	MERCEDES BENZ	E CLASS	DIESEL	Black	87700	SEDAN	2013-04-01	
6	VOLKSWAGEN	AMEO	DIESEL	Blue	70577	SEDAN	2017-06-01	
7	MARUTI SUZUKI	ERTIGA	PETROL	A Blue	76259	MPV	2013-07-01	
8	MARUTI SUZUKI	SWIFT	PETROL	Silver	85000	HATCHBACK	2015-02-01	
9	HYUNDAI	I10	PETROL	Red	77000	HATCHBACK	2008-05-01	
10	VOLKSWAGEN	AMEO	PETROL	Steel Grey	29416	SEDAN	2018-07-01	
11	MARUTI SUZUKI	CELERIO	PETROL	Red	64277	HATCHBACK	2017-09-01	
12	HYUNDAI	CRETA(2018-2019)	DIESEL	Red	51078	SUV	2018-09-01	
13	MARUTI SUZUKI	VITARA BREZZA	DIESEL	Red	66535	SUV	2017-04-01	
14	MAHINDRA	XUV500	DIESEL	Moondust Silver	58422	SUV	2016-06-01	
15	HONDA	JAZZ	PETROL	Orange	53377	HATCHBACK	2016-01-01	
16	HYUNDAI	GRAND I10	PETROL	Marine Blue	19365	HATCHBACK	2019-01-01	
17	RENAULT	DUSTER	PETROL	Brown	42773	SUV	2017-01-01	
18	NISSAN	MICRA ACTIVE	PETROL	Silver	17511	HATCHBACK	2015-11-01	
19	MARUTI SUZUKI	CELERIO	PETROL	Red	48042	HATCHBACK	2016-01-01	

In [260…

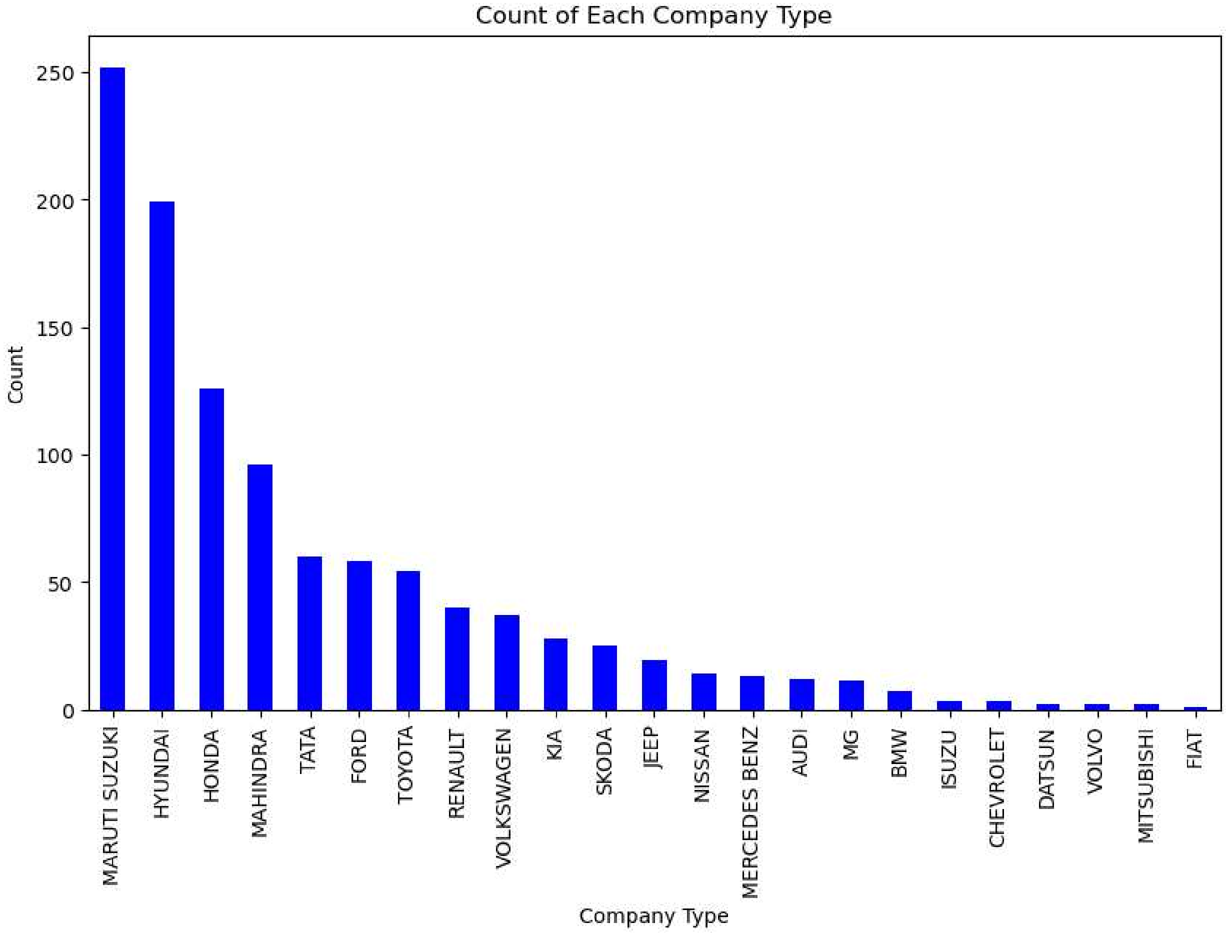
```
df.nunique()
```

Out[260]:

```
Company              23
Model               218
FuelType              5
Colour               76
Kilometer          1006
BodyStyle            10
ManufactureDate     162
ModelYear            17
Price               367
Owner                 4
DealerState          10
DealerName           57
City                 11
Warranty              2
QualityScore         43
dtype: int64
```

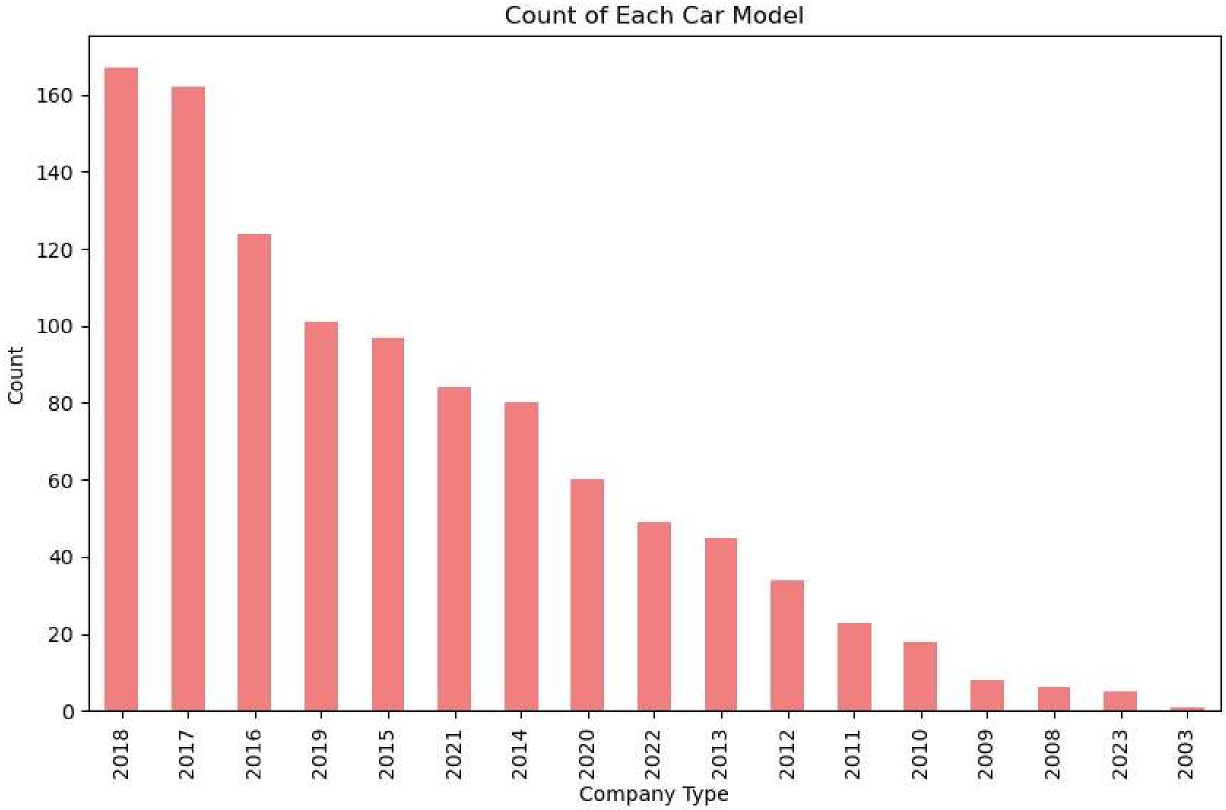
In [261…

```
company_counts = df['Company'].value_counts()

# Create a bar chart
mat.figure(figsize=(10, 6))
company_counts.plot(kind='bar', color='blue')
mat.title("Count of Each Company Type")
mat.xlabel("Company Type")
mat.ylabel("Count")
mat.show()
```



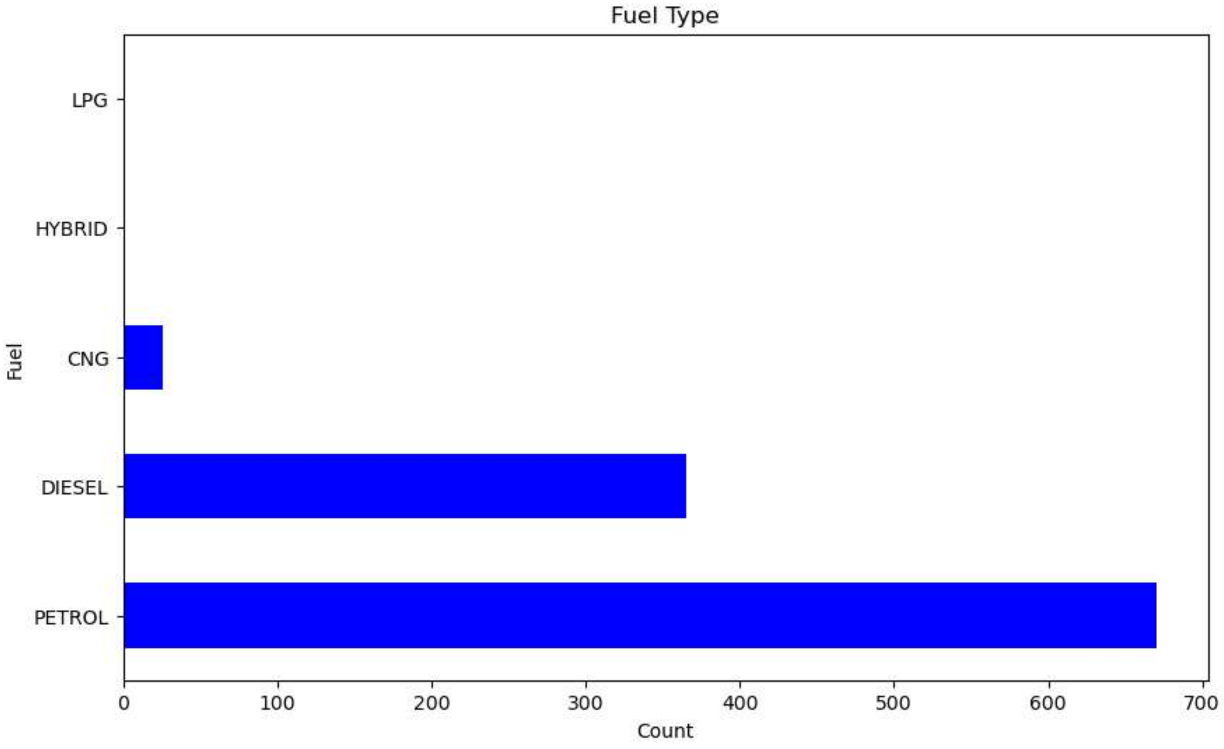
In [262…

```
company_counts = df['ModelYear'].value_counts()

# Create a bar chart
mat.figure(figsize=(10, 6))
company_counts.plot(kind='bar', color='lightcoral')
mat.title("Count of Each Car Model")
mat.xlabel("Company Type")
mat.ylabel("Count")
mat.show()
```



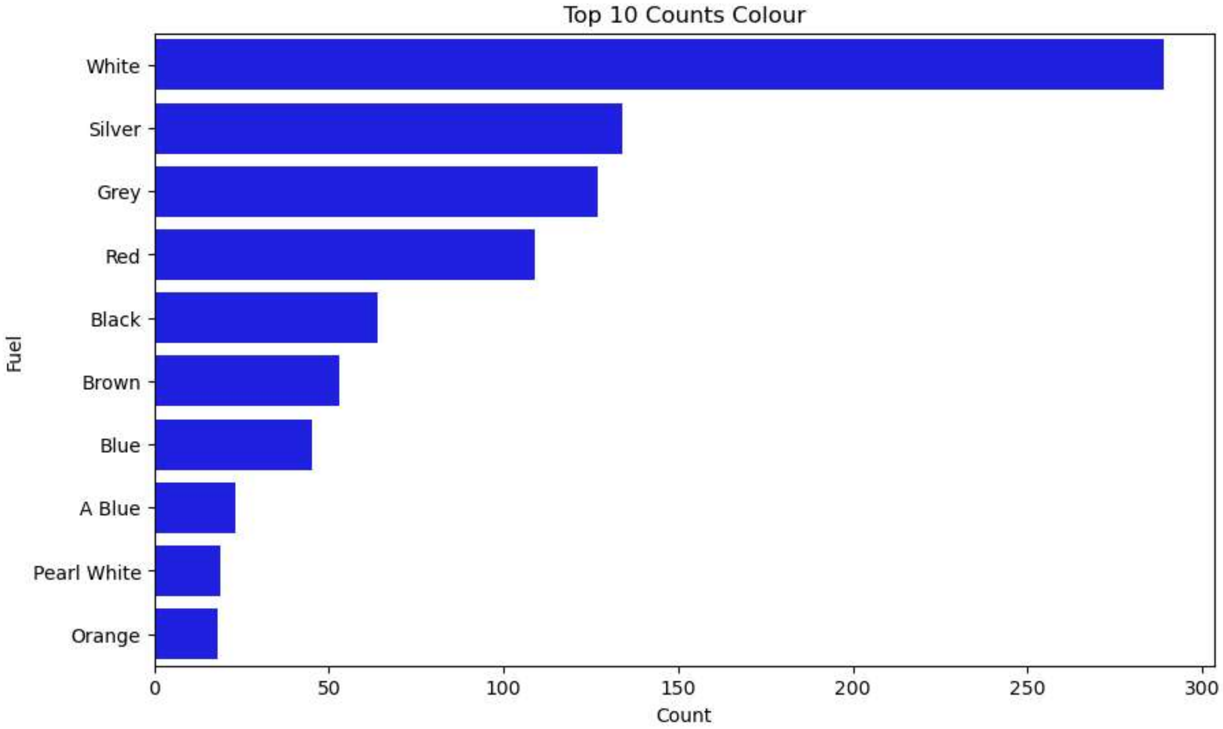
In [263…

```
company_counts = df['FuelType'].value_counts()

# Create a bar chart
mat.figure(figsize=(10, 6))
company_counts.plot(kind='barh', color='blue')
mat.title("Fuel Type")
mat.xlabel("Count")
mat.ylabel("Fuel")
mat.show()
```



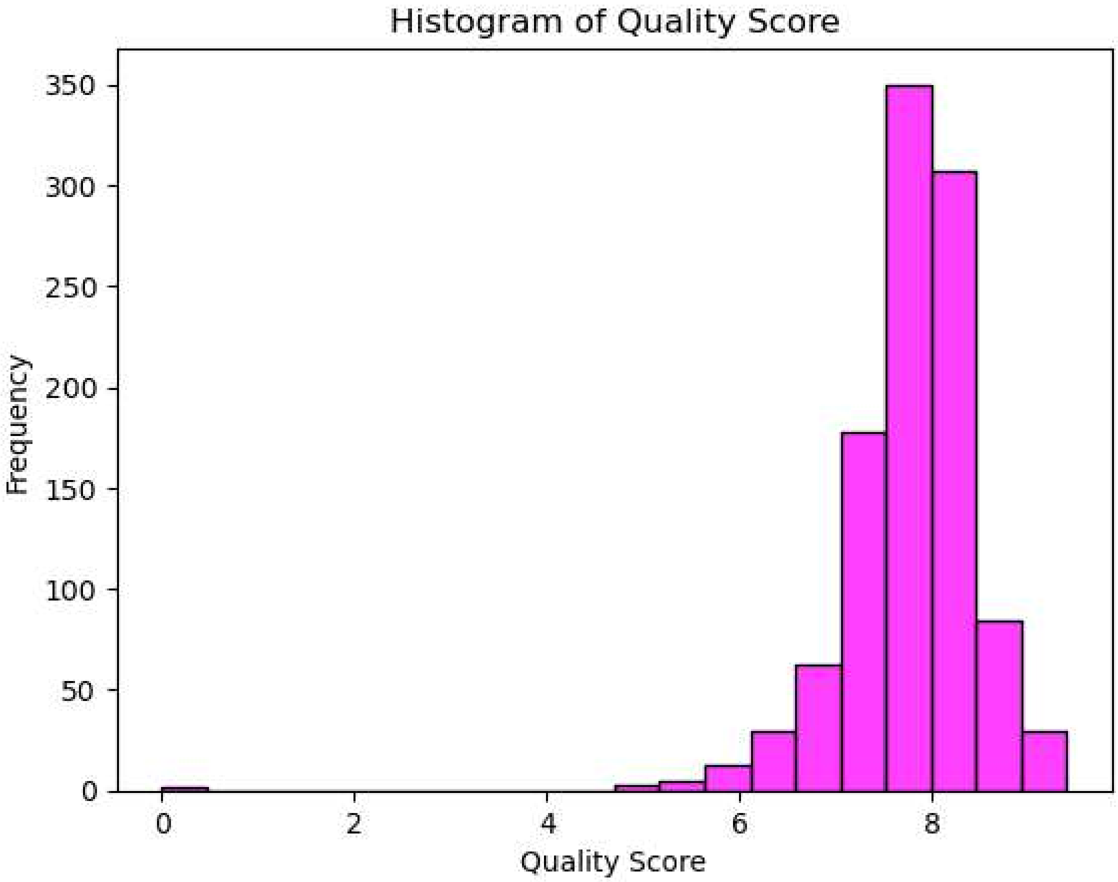
In [264…

```
# Create a figure
fig, ax = mat.subplots(figsize=(10, 6))

# Get the top 10 counts and their corresponding colors
top_10_counts = df['Colour'].value_counts().head(10)

# Create a horizontal bar chart for the top 10 counts
sea.barplot(x=top_10_counts, y=top_10_counts.index, color='blue', ax=ax)

# Set labels and title for the plot
ax.set_xlabel("Count")
ax.set_ylabel("Fuel")
ax.set_title("Top 10 Counts Colour")

# Show the plot
mat.show()
```



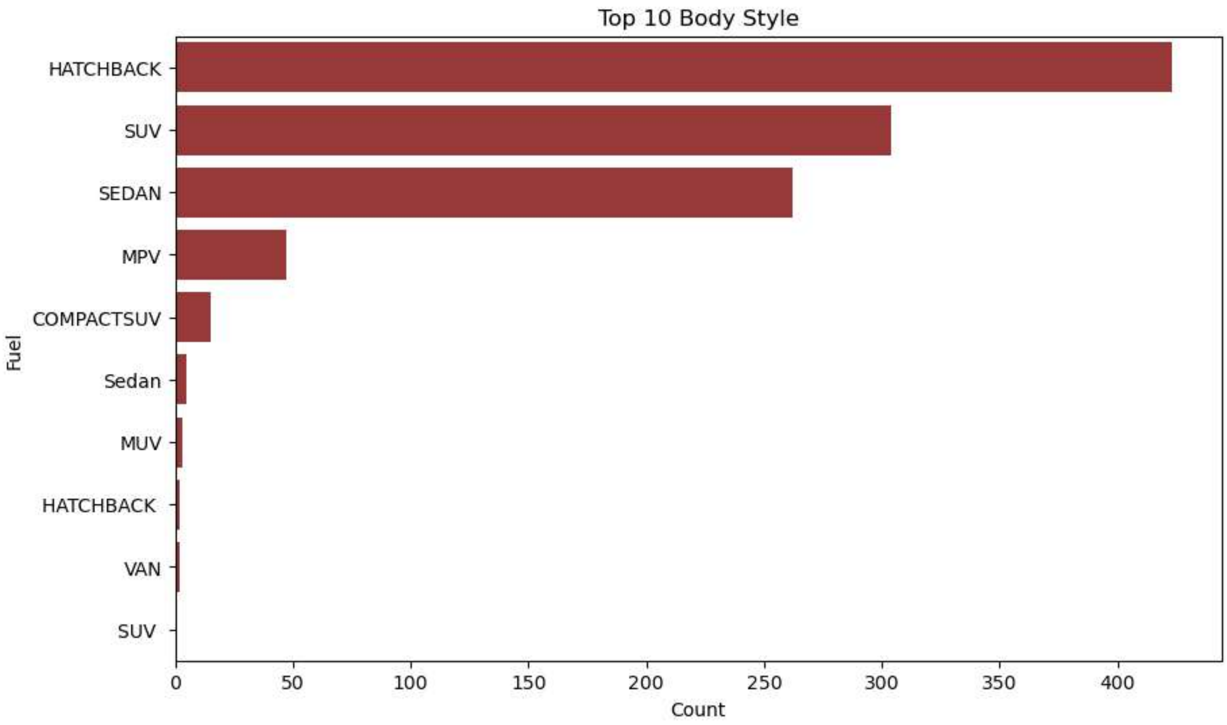
In [265…

```
sea.histplot(data=df, x='QualityScore', bins=20, color='magenta')

# Set labels and title
mat.xlabel('Quality Score')
mat.ylabel('Frequency')
mat.title('Histogram of Quality Score')

# Show the plot
mat.show()
```



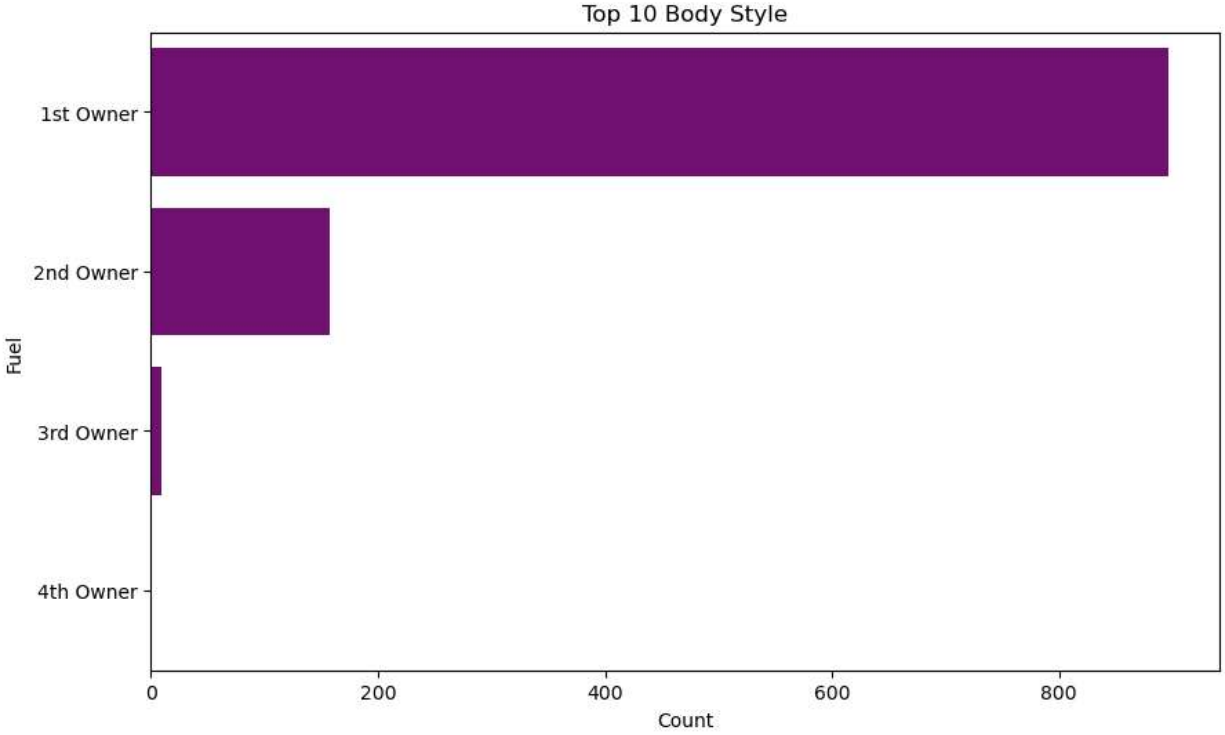
In [266…

```
# Create a figure
fig, ax = mat.subplots(figsize=(10, 6))

# Get the top 10 counts and their corresponding colors
top_10_counts = df['BodyStyle'].value_counts().head(10)

# Create a horizontal bar chart for the top 10 counts
sea.barplot(x=top_10_counts, y=top_10_counts.index, color='brown', ax=ax)

# Set labels and title for the plot
ax.set_xlabel("Count")
ax.set_ylabel("Fuel")
ax.set_title("Top 10 Body Style")

# Show the plot
mat.show()
```



In [267…

```
# Create a figure
fig, ax = mat.subplots(figsize=(10, 6))

# Get the top 10 counts and their corresponding colors
top_10_counts = df['Owner'].value_counts()

# Create a horizontal bar chart for the top 10 counts
sea.barplot(x=top_10_counts, y=top_10_counts.index, color='purple', ax=ax)

# Set labels and title for the plot
ax.set_xlabel("Count")
ax.set_ylabel("Fuel")
ax.set_title("Top 10 Body Style")

# Show the plot
mat.show()
```



In [268…

```
df.dtypes
```

Out[268]:

```
Company              object
Model                object
FuelType             object
Colour               object
Kilometer             int64
BodyStyle            object
ManufactureDate      object
ModelYear             int64
Price                object
Owner                object
DealerState          object
DealerName           object
City                 object
Warranty              int64
QualityScore        float64
dtype: object
```

In [269…

```
#LabelEncoder to convert all the non-numeric values to numeric
cols = df.columns[df.dtypes == 'object']

#Label encoder object
le = LabelEncoder()

#Label encoding the columns
for i in cols:
    le.fit(df[i])
    df[i] = le.transform(df[i])
```

In [270…

```
df.dtypes
```

Out[270]:

```
Company              int32
Model                int32
FuelType             int32
Colour               int32
Kilometer            int64
BodyStyle            int32
ManufactureDate      int32
ModelYear            int64
Price                int32
Owner                int32
DealerState          int32
DealerName           int32
City                 int32
Warranty             int64
QualityScore       float64
dtype: object
```

In [271…

```
#removing the outliers
z_score_threshold = 3

z_scores = num.abs((df[['FuelType', 'ManufactureDate', 'Warranty','Company']] - df[['F

df_no_outliers = df[(z_scores <= z_score_threshold).all(axis=1)]
```

In [272…

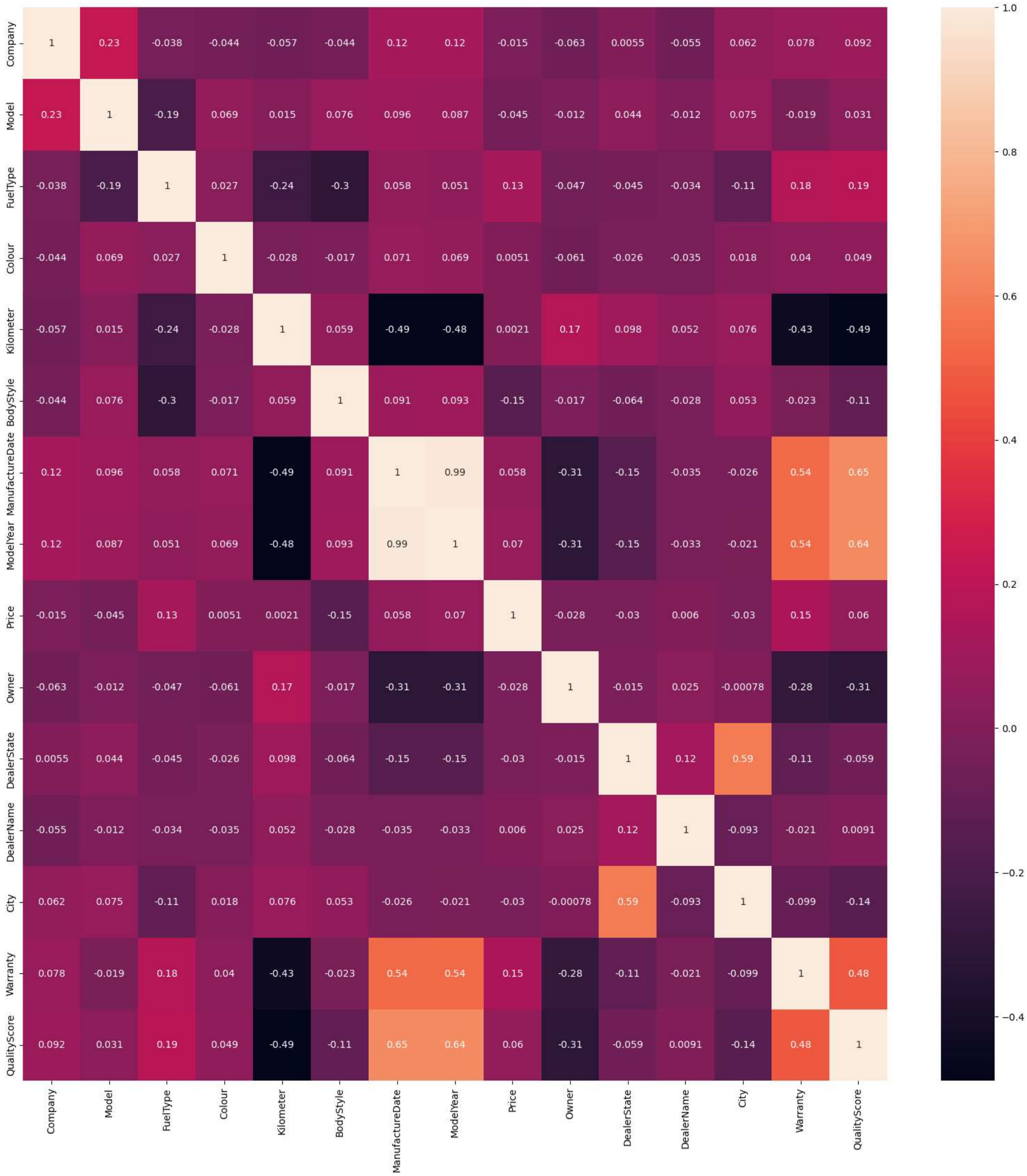
```
#Min_Max Scaler
Min_Max = MinMaxScaler()

df['FuelType'] = Min_Max.fit_transform(df['FuelType'].values.reshape(-1,1))
df['ManufactureDate']= Min_Max.fit_transform(df['ManufactureDate'].values.reshape(-1,1
df['Warranty']= Min_Max.fit_transform(df['Warranty'].values.reshape(-1,1))
df['Company']= Min_Max.fit_transform(df['Company'].values.reshape(-1,1))
```

In [273…

```
mat.figure(figsize=(20,20))
sea.heatmap(df.corr(), annot=True)
```

Out[273]:

```
<Axes: >
```



	Company	Model	FuelType	Colour	Kilometer	BodyStyle	ManufactureDate	ModelYear	Price	Owner	DealerState	DealerName	City	Warranty	QualityScore
Company	1	0.23	-0.038	-0.044	-0.057	-0.044	0.12	0.12	-0.015	-0.063	0.0055	-0.055	0.062	0.078	0.092
Model	0.23	1	-0.19	0.069	0.015	0.076	0.096	0.087	-0.045	-0.012	0.044	-0.012	0.075	-0.019	0.031
FuelType	-0.038	-0.19	1	0.027	-0.24	-0.3	0.058	0.051	0.13	-0.047	-0.045	-0.034	-0.11	0.18	0.19
Colour	-0.044	0.069	0.027	1	-0.028	-0.017	0.071	0.069	0.0051	-0.061	-0.026	-0.035	0.018	0.04	0.049
Kilometer	-0.057	0.015	-0.24	-0.028	1	0.059	-0.49	-0.48	0.0021	0.17	0.098	0.052	0.076	-0.43	-0.49
BodyStyle	-0.044	0.076	-0.3	-0.017	0.059	1	0.091	0.093	-0.15	-0.017	-0.064	-0.028	0.053	-0.023	-0.11
ManufactureDate	0.12	0.096	0.058	0.071	-0.49	0.091	1	0.99	0.058	-0.31	-0.15	-0.035	-0.026	0.54	0.65
ModelYear	0.12	0.087	0.051	0.069	-0.48	0.093	0.99	1	0.07	-0.31	-0.15	-0.033	-0.021	0.54	0.64
Price	-0.015	-0.045	0.13	0.0051	0.0021	-0.15	0.058	0.07	1	-0.028	-0.03	0.006	-0.03	0.15	0.06
Owner	-0.063	-0.012	-0.047	-0.061	0.17	-0.017	-0.31	-0.31	-0.028	1	-0.015	0.025	-0.00078	-0.28	-0.31
DealerState	0.0055	0.044	-0.045	-0.026	0.098	-0.064	-0.15	-0.15	-0.03	-0.015	1	0.12	0.59	-0.11	-0.059
DealerName	-0.055	-0.012	-0.034	-0.035	0.052	-0.028	-0.035	-0.033	0.006	0.025	0.12	1	-0.093	-0.021	0.0091
City	0.062	0.075	-0.11	0.018	0.076	0.053	-0.026	-0.021	-0.03	-0.00078	0.59	-0.093	1	-0.099	-0.14
Warranty	0.078	-0.019	0.18	0.04	-0.43	-0.023	0.54	0.54	0.15	-0.28	-0.11	-0.021	-0.099	1	0.48
QualityScore	0.092	0.031	0.19	0.049	-0.49	-0.11	0.65	0.64	0.06	-0.31	-0.059	0.0091	-0.14	0.48	1

In [274...

```
#define the training and learning model
X_train, X_test, y_train, y_test = train_test_split(df.drop(columns='Price'), df['Pric
```

In [275...

```
from sklearn.model_selection import GridSearchCV

#Random_forest_clasifier
rfc = RandomForestClassifier()
param_grid = {
    'max_depth': [2,4,6,8,10],
    'min_samples_leaf': [2,4,6,8,10],
    'min_samples_split': [2,4,6,8,10],
    'criterion': ['gini', 'entropy'],
    'random_state': [0,42]
}
```

```
#Grid Search Object with Random Forest Classifier
grid_search = GridSearchCV(estimator = rfc, param_grid = param_grid, cv = 3, n_jobs =

#Fitting the data
grid_search.fit(X_train, y_train)

#Best parameters
print(grid_search.best_params_)
```

```
Fitting 3 folds for each of 500 candidates, totalling 1500 fits
```

```
C:\Users\mohieldine\AppData\Local\anaconda3\Lib\site-packages\sklearn\model_selection
\_split.py:700: UserWarning: The least populated class in y has only 1 members, which
is less than n_splits=3.
  warnings.warn(
```

```
{'criterion': 'entropy', 'max_depth': 8, 'min_samples_leaf': 2, 'min_samples_split':
8, 'random_state': 42}
```

In [276…

```
#Random Forest Classifier accuracy
rfc = RandomForestClassifier(criterion='entropy', max_depth=10, min_samples_leaf=10, m

#Fitting the data
rfc.fit(X_train, y_train)

#Training accuracy
print('Training Accuracy: ', rfc.score(X_train, y_train))

#Predicting the values
r_pred = rfc.predict(X_test)
```

```
Training Accuracy:  0.7215041128084606
```

In [277…

```
from sklearn.model_selection import GridSearchCV

#Decision_Tree_Classifier
dtree  = DecisionTreeClassifier()
param_grid = {
    'max_depth': [2,4,6,8,10],
    'min_samples_leaf': [2,4,6,8,10],
    'min_samples_split': [2,4,6,8,10],
    'criterion': ['gini', 'entropy'],
    'random_state': [0,42]
}

#Grid Search Object with Decision Tree Classifier
grid_search = GridSearchCV(estimator = dtree, param_grid = param_grid, cv = 3, n_jobs

#Fitting the data
grid_search.fit(X_train, y_train)

#Best parameters
print(grid_search.best_params_)
```

```
Fitting 3 folds for each of 500 candidates, totalling 1500 fits
```

```
C:\Users\mohieldine\AppData\Local\anaconda3\Lib\site-packages\sklearn\model_selection
\_split.py:700: UserWarning: The least populated class in y has only 1 members, which
is less than n_splits=3.
  warnings.warn(
```

```
{'criterion': 'gini', 'max_depth': 4, 'min_samples_leaf': 2, 'min_samples_split': 2,
'random_state': 0}
```

In [278…

```
#Decision Tree Classifier accuracy
dtree  = DecisionTreeClassifier(criterion='gini', max_depth=6, min_samples_leaf=2, mir

#Fitting the data
dtree.fit(X_train, y_train)

#Training accuracy
print('Training Accuracy: ', dtree.score(X_train, y_train))

#Predicting the values
d_pred = dtree.predict(X_test)
```

```
Training Accuracy:  0.14336075205640422
```

In [279…

```
from sklearn.model_selection import GridSearchCV

#K Nearst Classifier
knn = KNeighborsClassifier()
param_grid = {
    'n_neighbors': [2,4,6,8,10],
    'weights': ['uniform', 'distance'],
    'algorithm': ['auto', 'ball_tree', 'kd_tree', 'brute']
}

#Grid Search Object with KNN Classifier
grid_search = GridSearchCV(estimator = knn, param_grid = param_grid, cv = 3, n_jobs =

#Fitting the data
grid_search.fit(X_train, y_train)

#Best parameters
print(grid_search.best_params_)
```

```
Fitting 3 folds for each of 40 candidates, totalling 120 fits
C:\Users\mohieldine\AppData\Local\anaconda3\Lib\site-packages\sklearn\model_selection
\_split.py:700: UserWarning: The least populated class in y has only 1 members, which
is less than n_splits=3.
  warnings.warn(
{'algorithm': 'auto', 'n_neighbors': 8, 'weights': 'uniform'}
```

In [280…

```
#KNN Classifier Acurracy
knn = KNeighborsClassifier(algorithm='ball_tree', n_neighbors=6, weights='uniform')

#Fitting the data
knn.fit(X_train, y_train)

#Training accuracy
print('Training Accuracy: ', knn.score(X_train, y_train))

#Predicting the values
k_pred = knn.predict(X_test)
```

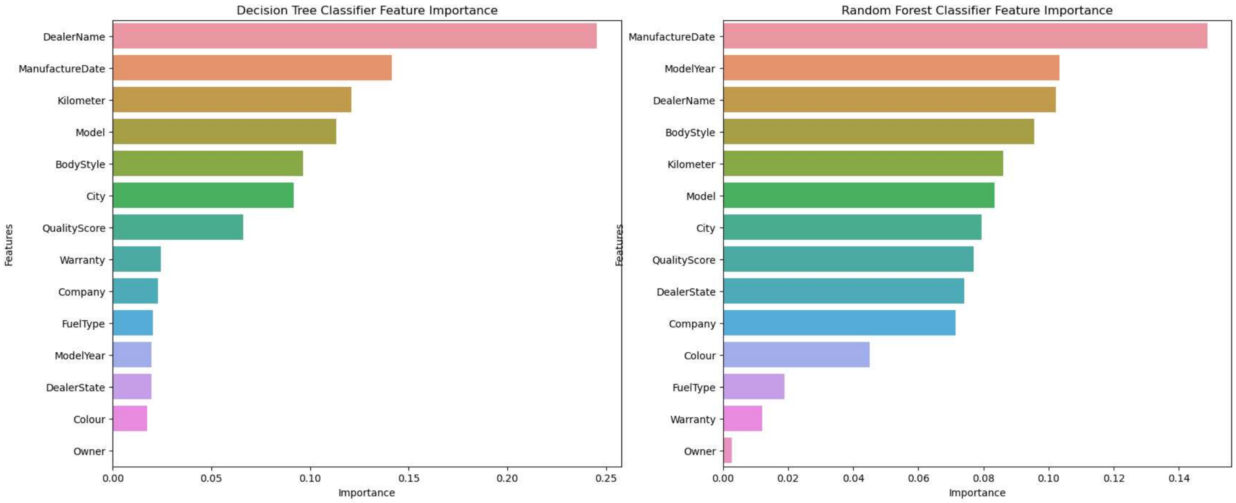
```
Training Accuracy:  0.18801410105757932
```

In [281…

```
fig, ax = mat.subplots(1, 2, figsize=(20, 8))
# Decision Tree Classifier Feature Importance
feature_df = pd.DataFrame({'Features': X_train.columns, 'Importance': dtree.feature_im
feature_df.sort_values('Importance', ascending=False, inplace=True)
sea.barplot(x = 'Importance', y = 'Features', data = feature_df, ax=ax[0]).set_title('
# Random Forest Classifier Feature Importance
feature_df = pd.DataFrame({'Features': X_train.columns, 'Importance': rfc.feature_impo
feature_df.sort_values('Importance', ascending=False, inplace=True)
sea.barplot(x = 'Importance', y = 'Features', data = feature_df, ax=ax[1]).set_title('
```

Out[281]:

```
Text(0.5, 1.0, 'Random Forest Classifier Feature Importance')
```



Decision Tree Classifier Feature Importance

Random Forest Classifier Feature Importance

In [ ]: