```
!pip install pydantic
!pip install PyYAML
!pip install jinja2
!pip install visions
!pip install htmlmin
!pip install phik
!pip install requests
!pip install tqdm
!pip install seaborn
!pip install multimethod
!pip install statsmodels
!pip install typeguard
!pip install imagehash
!pip install wordcloud
!pip install dacite
!pip install numba
```

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
```

```
! python --version
```

```
Python 3.10.12
```

```
import sys
!{sys.executable} -m pip install -U ydata-profiling
!jupyter nbextension enable --py widgetsnbextension
```

```
from google.colab import files
uploaded = files.upload()
```

```
Choose Files No file chosen  Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.
Saving used_cars.csv to used_cars.csv
```

```
data_file = "used_cars.csv"
df= pd.read_csv(data_file)
df
```



```
print(df.dtypes)
```

```
brand           object
model           object
model_year       int64
milage          object
fuel_type       object
engine          object
transmission    object
ext_col         object
int_col         object
accident        object
clean_title     object
price           object
dtype: object
```

354.0HP

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4009 entries, 0 to 4008
Data columns (total 12 columns):
 #   Column        Non-Null Count  Dtype 
---  ------        --------------  ----- 
 0   brand         4009 non-null   object
 1   model         4009 non-null   object
 2   model_year    4009 non-null   int64 
 3   milage        4009 non-null   object
 4   fuel_type     3839 non-null   object
 5   engine        4009 non-null   object
 6   transmission  4009 non-null   object
 7   ext_col       4009 non-null   object
 8   int_col       4009 non-null   object
 9   accident      3896 non-null   object
 10  clean_title   3413 non-null   object
 11  price         4009 non-null   object
dtypes: int64(1), object(11)
memory usage: 376.0+ KB
```

```
df['milage'] = df['milage'].str.replace(r'\D', '', regex=True)
df['milage'] = df['milage'].astype(float)
df['price'] = df['price'].str.replace(r'\D', '', regex=True)
df['price'] = df['price'].astype(float)
df
```

	brand	model	model_year	milage	fuel_type	engine	transmission	ext_
0	Ford	Utility Police Interceptor Base	2013	51000.0	E85 Flex Fuel	300.0HP 3.7L V6 Cylinder Engine Flex Fuel Capa...	6-Speed A/T	B
1	Hyundai	Palisade SEL	2021	34742.0	Gasoline	3.8L V6 24V GDI DOHC	8-Speed Automatic	Moon C
2	Lexus	RX 350 RX 350	2022	22372.0	Gasoline	3.5 Liter DOHC	Automatic	
3	INFINITI	Q50 Hybrid Sport	2015	88900.0	Hybrid	354.0HP 3.5L V6 Cylinder Engine Gas/Electric H...	7-Speed A/T	B
4	Audi	Q3 45 S line Premium Plus	2021	9835.0	Gasoline	2.0L I4 16V GDI DOHC Turbo	8-Speed Automatic	Gla W Met
...	...	...	...	...	...	...	...	

```
df.isna().sum()
#df.dtypes
```

```
brand             0
model             0
model_year        0
milage            0
fuel_type       170
engine            0
transmission      0
ext_col           0
int_col           0
accident        113
clean_title     596
price             0
dtype: int64
```

```
df['accident'] = df['accident'].replace({'At least 1 accident or damage reported' : 'Yes',
'None reported': 'No'})
df['clean_title'] = df['clean_title'].fillna('No')
#this last part is done by me
df['accident'] = df['accident'].fillna('No')
df
```

	brand	model	model_year	milage	fuel_type	engine	transmission	ext_
0	Ford	Utility Police Interceptor Base	2013	51000.0	E85 Flex Fuel	300.0HP 3.7L V6 Cylinder Engine Flex Fuel Capa...	6-Speed A/T	B
1	Hyundai	Palisade SEL	2021	34742.0	Gasoline	3.8L V6 24V GDI DOHC	8-Speed Automatic	Moon C
2	Lexus	RX 350 RX 350	2022	22372.0	Gasoline	3.5 Liter DOHC	Automatic	I
3	INFINITI	Q50 Hybrid Sport	2015	88900.0	Hybrid	354.0HP 3.5L V6 Cylinder Engine Gas/Electric H...	7-Speed A/T	B
4	Audi	Q3 45 S line Premium Plus	2021	9835.0	Gasoline	2.0L I4 16V GDI DOHC Turbo	8-Speed Automatic	Gl W Me
...	...	...	...	...	...	...	...	

```
sns.countplot(x = 'fuel_type', data = df)
```



```
<Axes: xlabel='fuel_type', ylabel='count'>
```

```
df['fuel_type'] = df['fuel_type'].fillna('Gasoline')
df
```

	brand	model	model_year	milage	fuel_type	engine	transmission	ext_
0	Ford	Utility Police Interceptor Base	2013	51000.0	E85 Flex Fuel	300.0HP 3.7L V6 Cylinder Engine Flex Fuel Capa...	6-Speed A/T	B
1	Hyundai	Palisade SEL	2021	34742.0	Gasoline	3.8L V6 24V GDI DOHC	8-Speed Automatic	Moon C
2	Lexus	RX 350 RX 350	2022	22372.0	Gasoline	3.5 Liter DOHC	Automatic	I
3	INFINITI	Q50 Hybrid Sport	2015	88900.0	Hybrid	354.0HP 3.5L V6 Cylinder Engine Gas/Electric H...	7-Speed A/T	B
4	Audi	Q3 45 S line Premium Plus	2021	9835.0	Gasoline	2.0L I4 16V GDI DOHC Turbo	8-Speed Automatic	Gl W Me
...	...	...	...	...	...	...	...	

```
df.isna().sum()
#df.dtypes
```

```
brand           0
model           0
model_year      0
milage          0
fuel_type       0
engine          0
transmission    0
ext_col         0
int_col         0
accident        0
clean_title     0
price           0
dtype: int64
```

```
df.dtypes
```

```
brand            object
model            object
model_year        int64
milage          float64
fuel_type        object
engine           object
transmission     object
ext_col          object
int_col          object
accident         object
clean_title      object
price           float64
dtype: object
```

```
Current_Year = 2023
df['age'] = Current_Year - df['model_year']
df['age'] = df['age'].astype(np.int64)
df
```

	brand	model	model_year	milage	fuel_type	engine	transmission	ext_
0	Ford	Utility Police Interceptor Base	2013	51000.0	E85 Flex Fuel	300.0HP 3.7L V6 Cylinder Engine Flex Fuel Capa...	6-Speed A/T	B
1	Hyundai	Palisade SEL	2021	34742.0	Gasoline	3.8L V6 24V GDI DOHC	8-Speed Automatic	Moon C
2	Lexus	RX 350 RX 350	2022	22372.0	Gasoline	3.5 Liter DOHC	Automatic	
3	INFINITI	Q50 Hybrid Sport	2015	88900.0	Hybrid	354.0HP 3.5L V6 Cylinder Engine Gas/Electric H...	7-Speed A/T	B
4	Audi	Q3 45 S line Premium Plus	2021	9835.0	Gasoline	2.0L I4 16V GDI DOHC Turbo	8-Speed Automatic	Gl W Me

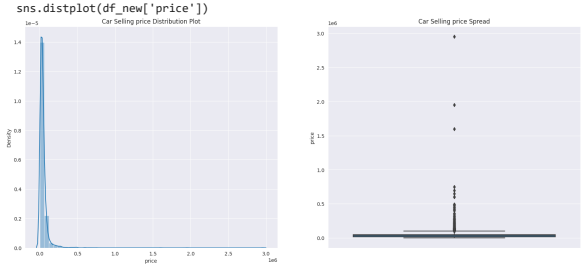
```
df_new = df.drop(['model_year'], axis=1)
df_new
```

	brand	model	milage	fuel_type	engine	transmission	ext_col	int_col
0	Ford	Utility Police Interceptor Base	51000.0	E85 Flex Fuel	300.0HP 3.7L V6 Cylinder Engine Flex Fuel Capa...	6-Speed A/T	Black	Black
1	Hyundai	Palisade SEL	34742.0	Gasoline	3.8L V6 24V GDI DOHC	8-Speed Automatic	Moonlight Cloud	Gray
2	Lexus	RX 350 RX 350	22372.0	Gasoline	3.5 Liter DOHC	Automatic	Blue	Black
3	INFINITI	Q50 Hybrid Sport	88900.0	Hybrid	354.0HP 3.5L V6 Cylinder Engine Gas/Electric H...	7-Speed A/T	Black	Black
4	Audi	Q3 45 S line Premium Plus	9835.0	Gasoline	2.0L I4 16V GDI DOHC Turbo	8-Speed Automatic	Glacier White Metallic	Black
...	...	...	...	...	...	...	...	...

```
plt.figure(figsize=(20,8))
plt.subplot(1,2,1)
plt.title('Car Selling price Distribution Plot')
sns.distplot(df_new['price'])
sns.set_style('darkgrid')

plt.subplot(1,2,2)
plt.title('Car Selling price Spread')
sns.boxplot(y=df_new['price'])
sns.set_style('darkgrid')

plt.show()
```



```
<ipython-input-17-69c6c010fec7>:4: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(df_new['price'])
```

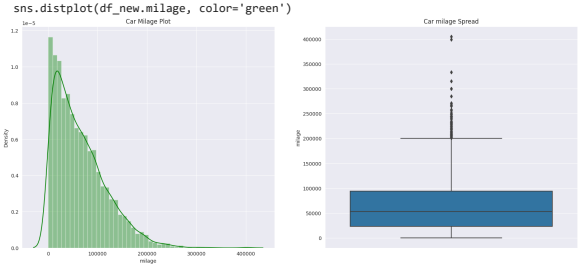



```
# plotting the target-age scatter graph
sns.scatterplot(data=df_new, x="age", y="price")
sns.set_style('darkgrid')
plt.title("Selling price by age", size=12)
plt.ylabel("Selling price (Thousand bucks)", size=10)
plt.xlabel("Age", size=10)
plt.show()
```


```
#Distribution
plt.figure(figsize=(20,8))
plt.subplot(1,2,1)
plt.title('Car Milage Plot')
sns.distplot(df_new.milage, color='green')
```

```
#Spread
plt.subplot(1,2,2)
plt.title('Car milage Spread')
sns.boxplot(y=df_new.milage)

plt.show()
```

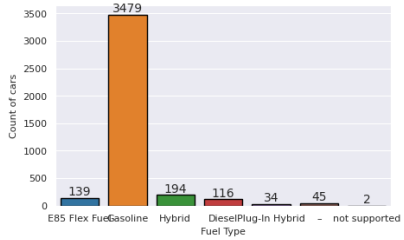
```
<ipython-input-19-cf6510de259a>:5: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(df_new.milage, color='green')
```



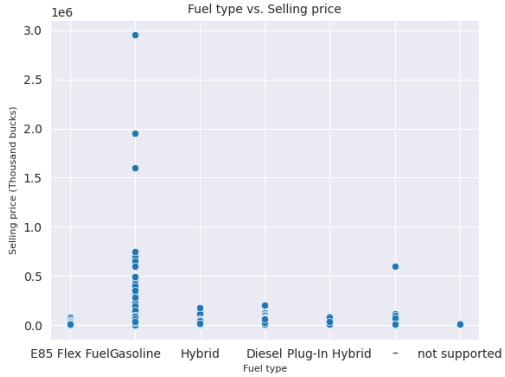

```
df_sym = pd.DataFrame(df_new['accident'].value_counts())
df_sym.plot.pie(subplots=True, labels = df_sym.index.values, autopct='%1.1f%%', fontsize=8)
# Unsquish the pie.
plt.gca().set_aspect('equal')
plt.show()
```



```
# Count of cars by fuel_type
plt.figure(figsize = (5, 3))
ax=sns.countplot(data=df_new, x=df.fuel_type, ec='black')
sns.set_style('darkgrid')
for cont in ax.containers:
    ax.bar_label(cont)
plt.ylabel('Count of cars', size=8)
plt.yticks(size=8)
plt.xlabel('Fuel Type', size=8)
plt.xticks(size=8)
plt.show()
```




```
# plotting the target-Fuel type scatter graph
sns.scatterplot(data=df_new, x="fuel_type", y="price")
sns.set_style('darkgrid')
plt.title("Fuel type vs. Selling price", size=10)
plt.ylabel("Selling price (Thousand bucks)", size=8)
plt.xlabel("Fuel type", size=8)
plt.show()
```

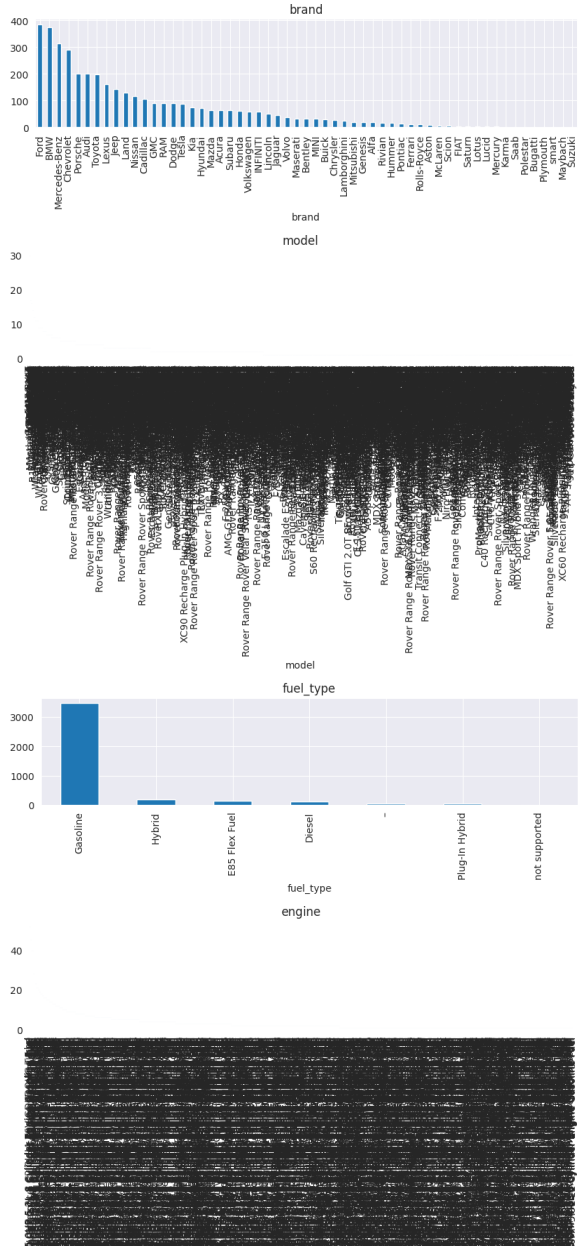


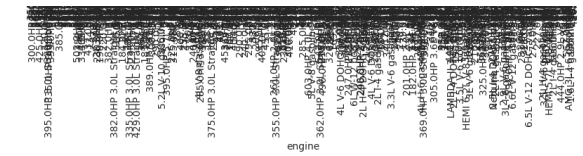

```
obj_cols = [col for col in df_new.columns if df_new[col].dtypes == 'O']
print('Number of Qualitative Variable: ', len(obj_cols))

def bar_charts(data, obj_cols):
    col_counter = 0
    data = df_new.copy()
    for col in obj_cols:
      data[col].value_counts().plot(kind = "bar",figsize=(10,2),fontsize=10)
      plt.xlabel(col)
      plt.title(col)
      plt.show()
```

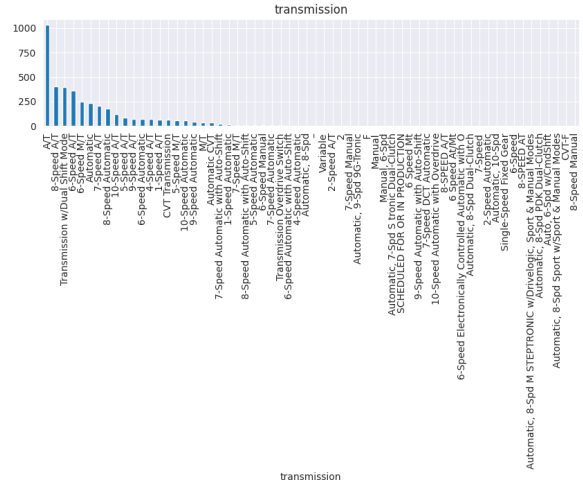
```
        col_counter += 1
    print(col_counter, "variables have been plotted")

bar_charts(df_new, obj_cols)
```

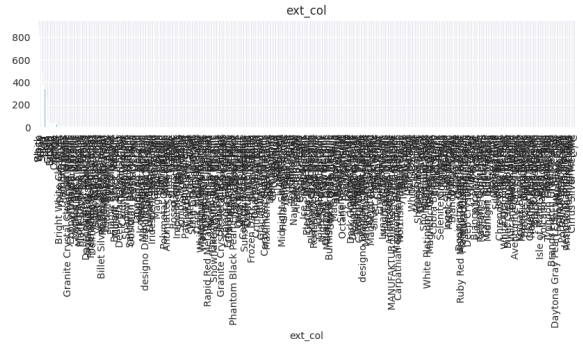


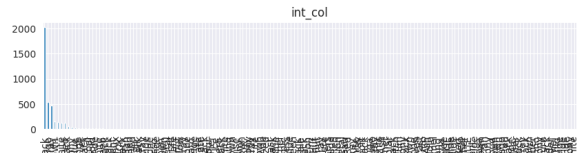
```
Number of Qualitative Variable:  9
```









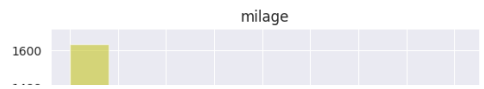



```
num_cols = [col for col in df_new.columns if df_new[col].dtypes != 'O']
print('Number of Numerical Variable: ', len(num_cols))


def hist_for_nums(data, numeric_cols):
    col_counter = 0
    data = data.copy()
```



```
    for col in numeric_cols:
        data[col].plot.hist(alpha=0.5, color='y')
        plt.xlabel(col)
        plt.title(col)
        plt.show()
        col_counter += 1
    print(col_counter, "variables have been plotted")
hist_for_nums(df_new, num_cols)
```

```
Number of Numerical Variable:  3
```




```
import ydata_profiling
from ydata_profiling import ProfileReport
profile = ProfileReport(df)
```

```
profile
```

Summarize dataset: 38/38 [00:08<00:00, 2.96it/s, 100% Completed]

Generate report structure: 100% 1/1 [00:12<00:00, 12.98s/it]

Render HTML: 100% 1/1 [00:01<00:00, 1.30s/it]

Overview

Dataset statistics

Number of variables	13
Number of observations	4009
Missing cells	0
Missing cells (%)	0.0%
Duplicate rows	0
Duplicate rows (%)	0.0%
Total size in memory	407.3 KiB
Average record size in memory	104.0 B

Variable types

Text	6
Numeric	4
Categorical	1
Boolean	2

Alerts

`model_year` is highly overall correlated with `milage` and 2 other fields (milage, price, age) High correlation

```
from sklearn import preprocessing
le = preprocessing.LabelEncoder()
for x in df_new.columns:
    if df_new[x].dtypes=='object':
        df_new[x]=le.fit_transform(df_new[x].astype(str))
corr = df_new.corr()
corr
```


	brand	model	milage	fuel_type	engine	transmission	ext_col
brand	1.000000	-0.070170	-0.012389	0.033300	-0.066116	-0.005099	-0.002001
model	-0.070170	1.000000	0.031513	0.004079	-0.037443	-0.024244	-0.008342
milage	-0.012389	0.031513	1.000000	-0.096195	-0.227913	-0.043796	0.000891
fuel_type	0.033300	0.004079	-0.096195	1.000000	0.080890	0.094140	-0.010056
engine	-0.066116	-0.037443	-0.227913	0.080890	1.000000	-0.011988	-0.037665
transmission	-0.005099	-0.024244	-0.043796	0.094140	-0.011988	1.000000	0.001548
ext_col	-0.002001	-0.008342	0.000891	-0.010056	-0.037665	0.001548	1.000000
int_col	0.008545	0.040801	-0.051394	0.013986	0.023628	-0.030224	0.085077

```
#df.drop(['model'], axis = 1)
df_new
```

	brand	model	milage	fuel_type	engine	transmission	ext_col	int_col	accident
0	14	1743	51000.0	1	581	16	29	14	1
1	19	1182	34742.0	2	566	32	185	71	1
2	27	1325	22372.0	2	541	40	38	14	0
3	20	1242	88900.0	3	724	23	29	14	0
4	3	1225	9835.0	2	200	32	120	14	0
...	...	...	...	...	...	...	...	...	...
4004	5	484	714.0	2	1060	33	50	75	0
4005	3	1464	10900.0	2	714	59	29	14	0
4006	43	1677	2116.0	2	1133	40	29	14	0
4007	14	666	33000.0	2	917	38	38	14	0
4008	4	1790	43000.0	2	356	38	128	31	1

4009 rows × 13 columns

```
X = df_new.iloc[:, list(range(10)) + [-1]]
y = df_new.iloc[:, -2]


X
#y


from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size= 0.2, random_state= 1)


print(X_train.shape)
print(X_test.shape)
print(y_train.shape)
print(y_test.shape)
```

```
(3207, 11)
(802, 11)
(3207,)
(802,)
```

```
from sklearn.linear_model import LinearRegression
# Create an instance of the LinearRegression class
reg = LinearRegression()

# Fit the model to the data
reg.fit(X_train, y_train)
score_LR = reg.score(X_test, y_test)

print(score_LR)
```

```
0.31915467345097737
```

```
# Print the coefficients and intercept of the model
print(reg.coef_)
print('Intercept: ', reg.intercept_)
```

```
[ 2.11754287e+02 -1.51749000e+00 -3.82413150e-01 -4.68630937e+03
  6.12280708e+01  1.86939042e+02  5.33937334e+00  6.73130085e+01
 -2.26895985e+03 -3.41477166e+03  1.45479648e+02]
Intercept:  31951.619637876604
```

Double-click (or enter) to edit

```
y_pred = reg.predict(X_test)
```

```
import math
from sklearn.metrics import mean_absolute_error,mean_squared_error, r2_score

mae = mean_absolute_error(y_true=y_test,y_pred=y_pred)
#squared True returns MSE value, False returns RMSE value.
mse = mean_squared_error(y_true=y_test,y_pred=y_pred) #default=True
rmse = mean_squared_error(y_true=y_test,y_pred=y_pred,squared=False)
#rmse = math.sqrt(mse)

print("MAE:",mae)
print("MSE:",mse)
print("RMSE:",rmse)
```

```
MAE: 22922.87178300498
MSE: 1825516368.7970462
RMSE: 42726.0619387868
```

```
#Cross avlidation for Linear Regression

from sklearn.model_selection import KFold
from sklearn.model_selection import cross_val_score
from numpy import mean
from numpy import absolute
from numpy import sqrt
```

```
#Cross avlidation for Linear Regression
#define cross-validation method to use
cv = KFold(n_splits=10, random_state=1, shuffle=True)

#use k-fold CV to evaluate linear Regression model
scores = cross_val_score(reg, X, y, scoring='neg_mean_absolute_error',
                         cv=cv, n_jobs=-1)
score = cross_val_score(reg, X, y, scoring='r2',
                         cv=cv, n_jobs=-1)

print(mean(score))

#view mean absolute error
print(mean(absolute(scores)))
#view RMSE
print(sqrt(mean(absolute(scores))))
```

```
0.24238868951613926
23366.22362134199
152.86014399228463
```

```
#Random Forest Regression
from sklearn.ensemble import RandomForestRegressor
RF_regressor = RandomForestRegressor(n_estimators= 10, random_state= 0)
RF_regressor.fit(X_train, y_train)
```

```
RandomForestRegressor
RandomForestRegressor(n_estimators=10, random_state=0)
```

```
#Predicting the target values of the test set
y_pred_RF = RF_regressor.predict(X_test)
score_RF = RF_regressor.score(X_test, y_test)
print(score_RF)
#MSE
```



```
mse = float(mean_squared_error(y_test, y_pred_RF))
rmse = float(mean_squared_error(y_test, y_pred_RF, squared = False))
print("MSE: ", mse)
# RMSE (Root Mean Square Error)
#rmse = float(format(np.sqrt(mean_squared_error(y_test, y_pred_RF)), '.3f'))
print("\nRMSE: ", rmse)
```

```
0.41447950921795773
MSE:  1569926675.720137

RMSE:  39622.30023257278
```

```
# Random forets with Cross Validation
scores_RF = cross_val_score(RF_regressor, X, y, scoring='neg_mean_absolute_error',
                            cv=10)
score_RF = cross_val_score(RF_regressor, X, y, scoring='r2',
                            cv=cv, n_jobs=-1)
print(mean(score_RF))
#view mean absolute error
print(mean(absolute(scores_RF)))
#view RMSE
print(sqrt(mean(absolute(scores_RF))))
```

```
0.3067158631451061
15954.262232418954
126.31018261572957
```

```
#KNN Regression
from sklearn.neighbors import KNeighborsRegressor

# Instance and fit
knn_model = KNeighborsRegressor(n_neighbors=5)
knn_model.fit(X_train, y_train)

# Score
score_knn = knn_model.score(X_test, y_test)
print(score_knn)
```

```
-0.10260211256746499
```

```
preds = knn_model.predict(X_test)

# Performance
performance = pd.DataFrame({ 'True Value': y_test,
                             'Prediction': preds,
                             'Error': y_test - preds})
# View
performance
```

	True Value	Prediction	Error
870	36500.0	27359.8	9140.2
929	14900.0	17899.2	-2999.2
1670	45985.0	33550.0	12435.0
701	119500.0	75495.6	44004.4
2308	30798.0	54829.4	-24031.4
...	...	...	...
2322	45000.0	89960.0	-44960.0
2543	41599.0	47280.0	-5681.0
2887	14500.0	15239.8	-739.8
2377	34000.0	31761.0	2239.0
2340	29000.0	18479.6	10520.4

802 rows × 3 columns

```
MSE = mean_squared_error(y_test, preds)
RMSE = mean_squared_error(y_test, preds, squared=False)
```

```
print('MSE:', MSE)
print('RMSE:', RMSE)
```

```
MSE: 2956351650.329177
RMSE: 54372.34269671647
```

```
from sklearn.model_selection import cross_val_score
#import numpy as np
#create a new KNN model
knn_scores = cross_val_score(knn_model, X, y, scoring='neg_mean_absolute_error',
                             cv=10)
knn_score = cross_val_score(knn_model, X, y, scoring='r2',
                            cv=10)

print(mean(knn_score))
#print('knn_scores mean:{}'.format(np.mean(cv_scores)))
#view mean absolute error
print(mean(absolute(knn_scores)))
#view RMSE
print(sqrt(mean(absolute(knn_scores))))
```

```
0.011950177812711682
22679.665068329174
150.59769277226385
```

```
#model Comparison
pd.DataFrame({'Linear Regression':[score_LR],
              'Random forest Regression': [score_RF],
              'KNN Regression': [score_knn]})
```

	Linear Regression	Random forest Regression	KNN Regression
0	0.319155	0.41448	-0.102602