Car Price Prediction

```
import pandas as pd
```

```
df = pd.read_csv('dataset/car data.csv')
```

```
df.head()
```

	Car_Name	Year	Selling_Price	Present_Price	Kms_Driven	Fuel_Type	Seller_Type	Transmission	Owner
0	ritz	2014	3.35	5.59	27000	Petrol	Dealer	Manual	0
1	sx4	2013	4.75	9.54	43000	Diesel	Dealer	Manual	0
2	ciaz	2017	7.25	9.85	6900	Petrol	Dealer	Manual	0
3	wagon r	2011	2.85	4.15	5200	Petrol	Dealer	Manual	0
4	swift	2014	4.60	6.87	42450	Diesel	Dealer	Manual	0

```
df.shape
```

```
(301, 9)
```

```
print(df['Seller_Type'].unique())
print(df['Transmission'].unique())
print(df['Owner'].unique())
```

```
['Dealer' 'Individual']
['Manual' 'Automatic']
[0 1 3]
```

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

```
Car_Name         0
Year             0
Selling_Price    0
Present_Price    0
Kms_Driven       0
Fuel_Type        0
Seller_Type      0
Transmission     0
Owner            0
dtype: int64
```

```
df.columns
```

```
Index(['Car_Name', 'Year', 'Selling_Price', 'Present_Price', 'Kms_Driven',
       'Fuel_Type', 'Seller_Type', 'Transmission', 'Owner'],
      dtype='object')
```

```
final_dataset = df[['Year', 'Selling_Price', 'Present_Price', 'Kms_Driven',
       'Fuel_Type', 'Seller_Type', 'Transmission', 'Owner']]
```

```
final_dataset.head()
```

	Year	Selling_Price	Present_Price	Kms_Driven	Fuel_Type	Seller_Type	Transmission	Owner
0	2014	3.35	5.59	27000	Petrol	Dealer	Manual	0
1	2013	4.75	9.54	43000	Diesel	Dealer	Manual	0
2	2017	7.25	9.85	6900	Petrol	Dealer	Manual	0
3	2011	2.85	4.15	5200	Petrol	Dealer	Manual	0
4	2014	4.60	6.87	42450	Diesel	Dealer	Manual	0

```
final_dataset['Current_Year'] = 2021
```

```
final_dataset.head()
```

	Year	Selling_Price	Present_Price	Kms_Driven	Fuel_Type	Seller_Type	Transmission	Owner	Current_Year
0	2014	3.35	5.59	27000	Petrol	Dealer	Manual	0	2021
1	2013	4.75	9.54	43000	Diesel	Dealer	Manual	0	2021
2	2017	7.25	9.85	6900	Petrol	Dealer	Manual	0	2021
3	2011	2.85	4.15	5200	Petrol	Dealer	Manual	0	2021
4	2014	4.60	6.87	42450	Diesel	Dealer	Manual	0	2021

```
final_dataset['Age'] = final_dataset['Current_Year']-final_dataset['Year']
```

```
final_dataset.head()
```

	Year	Selling_Price	Present_Price	Kms_Driven	Fuel_Type	Seller_Type	Transmission	Owner	Current_Year	Age
0	2014	3.35	5.59	27000	Petrol	Dealer	Manual	0	2021	7
1	2013	4.75	9.54	43000	Diesel	Dealer	Manual	0	2021	8
2	2017	7.25	9.85	6900	Petrol	Dealer	Manual	0	2021	4
3	2011	2.85	4.15	5200	Petrol	Dealer	Manual	0	2021	10
4	2014	4.60	6.87	42450	Diesel	Dealer	Manual	0	2021	7

```
final_dataset.drop(['Year'],axis=1,inplace=True)
#drops the column labelled as "year" and doesnt return a copy as inplace = true. and axis = 1 represents columns
```

```
final_dataset.drop(['Current_Year'],axis=1,inplace=True)
#drops the column labelled as "Current_year" and doesnt return a copy as inplace = true. and axis = 1 represents columns
```

```
final_dataset.head()
```

	Selling_Price	Present_Price	Kms_Driven	Fuel_Type	Seller_Type	Transmission	Owner	Age
0	3.35	5.59	27000	Petrol	Dealer	Manual	0	7
1	4.75	9.54	43000	Diesel	Dealer	Manual	0	8
2	7.25	9.85	6900	Petrol	Dealer	Manual	0	4
3	2.85	4.15	5200	Petrol	Dealer	Manual	0	10
4	4.60	6.87	42450	Diesel	Dealer	Manual	0	7

```
print(df['Fuel_Type'].unique())
```

```
['Petrol' 'Diesel' 'CNG']
```

```
final_dataset=pd.get_dummies(final_dataset,drop_first=True)
#removes multiple columns of the dataset as some column contain the same
#information because the original column could assume a binary value.
```

```
final_dataset.head()
```

	Selling_Price	Present_Price	Kms_Driven	Owner	Age	Fuel_Type_Diesel	Fuel_Type_Petrol	Seller_Type_Individual	Tr
0	3.35	5.59	27000	0	7	False	True	False	
1	4.75	9.54	43000	0	8	True	False	False	
2	7.25	9.85	6900	0	4	False	True	False	
3	2.85	4.15	5200	0	10	False	True	False	
4	4.60	6.87	42450	0	7	True	False	False	

```
final_dataset
```

	Selling_Price	Present_Price	Kms_Driven	Owner	Age	Fuel_Type_Diesel	Fuel_Type_Petrol	Seller_Type_Individual
0	3.35	5.59	27000	0	7	False	True	False
1	4.75	9.54	43000	0	8	True	False	False
2	7.25	9.85	6900	0	4	False	True	False
3	2.85	4.15	5200	0	10	False	True	False
4	4.60	6.87	42450	0	7	True	False	False
...	...	...	...	...	...	...	...	...
296	9.50	11.60	33988	0	5	True	False	False
297	4.00	5.90	60000	0	6	False	True	False
298	3.35	11.00	87934	0	12	False	True	False
299	11.50	12.50	9000	0	4	True	False	False
300	5.30	5.90	5464	0	5	False	True	False

301 rows × 9 columns

```
final_dataset.corr(method ='pearson')
#to find the pairwise correlation of all columns in the dataframe
```

	Selling_Price	Present_Price	Kms_Driven	Owner	Age	Fuel_Type_Diesel	Fuel_Type_Petrol
Selling_Price	1.000000	0.878983	0.029187	-0.088344	-0.236141	0.552339	-0.540571
Present_Price	0.878983	1.000000	0.203647	0.008057	0.047584	0.473306	-0.465244
Kms_Driven	0.029187	0.203647	1.000000	0.089216	0.524342	0.172515	-0.172874
Owner	-0.088344	0.008057	0.089216	1.000000	0.182104	-0.053469	0.055687
Age	-0.236141	0.047584	0.524342	0.182104	1.000000	-0.064315	0.059959
Fuel_Type_Diesel	0.552339	0.473306	0.172515	-0.053469	-0.064315	1.000000	-0.979648
Fuel_Type_Petrol	-0.540571	-0.465244	-0.172874	0.055687	0.059959	-0.979648	1.000000
Seller_Type_Individual	-0.550724	-0.512030	-0.101419	0.124269	0.039896	-0.350467	0.358321
Transmission_Manual	-0.367128	-0.348715	-0.162510	-0.050316	-0.000394	-0.098643	0.091013

```
import matplotlib.pyplot as plt
%matplotlib inline
#to display the plot directly below the code cell.


corrmat = final_dataset.corr(method='pearson')


corrmat.index
```

```
Index(['Selling_Price', 'Present_Price', 'Kms_Driven', 'Owner', 'Age',
       'Fuel_Type_Diesel', 'Fuel_Type_Petrol', 'Seller_Type_Individual',
       'Transmission_Manual'],
      dtype='object')
```

```
final_dataset
```

	Selling_Price	Present_Price	Kms_Driven	Owner	Age	Fuel_Type_Diesel	Fuel_Type_Petrol	Seller_Type_Individual
0	3.35	5.59	27000	0	7	False	True	False
1	4.75	9.54	43000	0	8	True	False	False
2	7.25	9.85	6900	0	4	False	True	False
3	2.85	4.15	5200	0	10	False	True	False
4	4.60	6.87	42450	0	7	True	False	False
...	...	...	...	...	...	...	...	...
296	9.50	11.60	33988	0	5	True	False	False
297	4.00	5.90	60000	0	6	False	True	False
298	3.35	11.00	87934	0	12	False	True	False
299	11.50	12.50	9000	0	4	True	False	False
300	5.30	5.90	5464	0	5	False	True	False

301 rows × 9 columns

```
final_dataset.iloc[:,0]
```

```
0       3.35
1       4.75
2       7.25
3       2.85
4       4.60
       ...
296     9.50
297     4.00
298     3.35
299    11.50
300     5.30
Name: Selling_Price, Length: 301, dtype: float64
```

```
X= final_dataset.iloc[:,1:]
#slicing the dataset and reomoving the selling price for training the model
Y = final_dataset.iloc[:,0]
#storing the selling price for checking..as this is the value to be predicted
```

```
final_dataset.head()
```

	Selling_Price	Present_Price	Kms_Driven	Owner	Age	Fuel_Type_Diesel	Fuel_Type_Petrol	Seller_Type_Individual
0	3.35	5.59	27000	0	7	False	True	False
1	4.75	9.54	43000	0	8	True	False	False
2	7.25	9.85	6900	0	4	False	True	False
3	2.85	4.15	5200	0	10	False	True	False
4	4.60	6.87	42450	0	7	True	False	False

```
X.head()
```

	Present_Price	Kms_Driven	Owner	Age	Fuel_Type_Diesel	Fuel_Type_Petrol	Seller_Type_Individual	Transmission_Manu
0	5.59	27000	0	7	False	True	False	Tr
1	9.54	43000	0	8	True	False	False	Tr
2	9.85	6900	0	4	False	True	False	Tr
3	4.15	5200	0	10	False	True	False	Tr
4	6.87	42450	0	7	True	False	False	Tr

```
Y.head()
```

```
0    3.35
1    4.75
```

```
2    7.25
3    2.85
4    4.60
Name: Selling_Price, dtype: float64
```

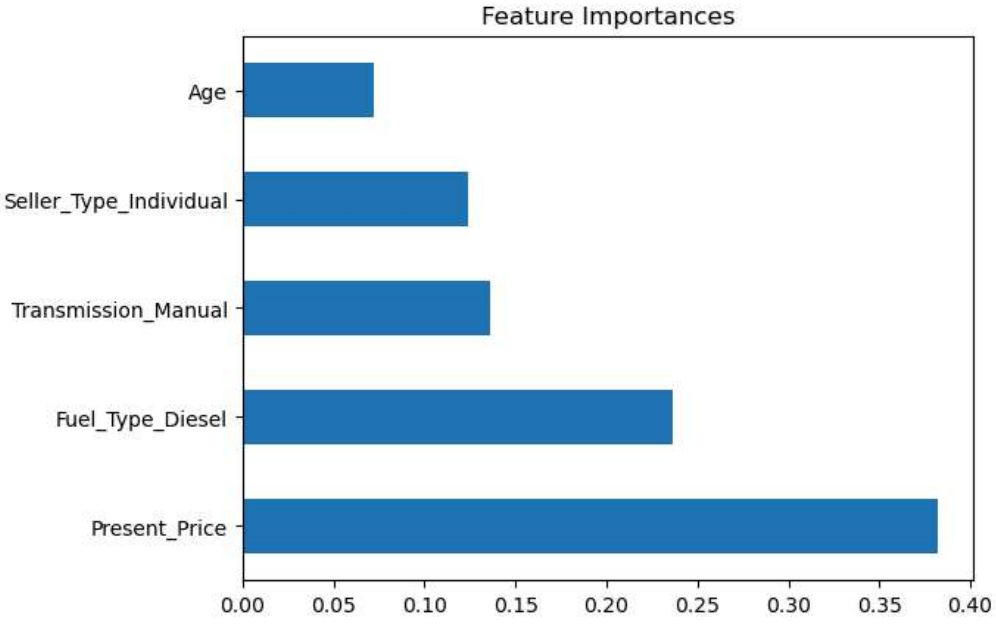
```
from sklearn.ensemble import ExtraTreesRegressor
#in ensemble predictions of several base estimators are built in with a given learning algorithm.
#we used ExtraTreesRegressor
model = ExtraTreesRegressor()
#This class implements a meta estimator that fits a number of randomized decision trees
#on various sub-samples of the dataset and uses averaging to improve the
#predictive accuracy and control over-fitting.
model.fit(X,Y)
```

```
▾ ExtraTreesRegressor
ExtraTreesRegressor()
```

```
print(model.feature_importances_)
#shows the feature importance that contribute to the selling price feature
```

```
[0.38232124 0.04238622 0.00042522 0.07216019 0.23654898 0.00605394
 0.12417712 0.13592709]
```

```
#plot graph of feature importances for better visualization
feat_importances = pd.Series(model.feature_importances_, index=X.columns)
feat_importances.nlargest(5).plot(kind='barh')
plt.title('Feature Importances')
plt.show()
```




```
from sklearn.model_selection import train_test_split #class to divide the data into train and validation set

X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=0.2, random_state=0)
#we divide the data into 2 parts :- 80% train and 20% test data
# and random_state is used to guarantee that same sequence of
#random numbers are generated each time you run the code.
#And unless there is some other randomness present in the process,
#the results produced will be same as always.
```

```
import numpy as np

# Reshape the input data for CNN
X_train_cnn = np.expand_dims(X_train.values, axis=2)
X_test_cnn = np.expand_dims(X_test.values, axis=2)

from tensorflow.keras.layers import Conv1D, MaxPooling1D, Flatten


# Define the CNN model
cnn_model = Sequential([
    Conv1D(64, 3, activation='relu', input_shape=(X_train_cnn.shape[1], X_train_cnn.shape[2])),
    MaxPooling1D(2),
    Conv1D(32, 3, activation='relu'),
    MaxPooling1D(2),
    Flatten(),
    Dense(64, activation='relu'),
    Dropout(0.5),
    Dense(32, activation='relu'),
    Dense(4, activation='softmax')  # Assuming 4 classes in 'class' column
], name='Improved_CNN_Model')

# Compile the model
cnn_model.compile(optimizer='adam', loss='sparse_categorical_crossentropy', metrics=['accuracy'])

cnn_model.summary()
```

```
---------------------------------------------------------------------------
ModuleNotFoundError                       Traceback (most recent call last)
Cell In[36], line 7
      4 X_train_cnn = np.expand_dims(X_train.values, axis=2)
      5 X_test_cnn = np.expand_dims(X_test.values, axis=2)
----> 7 from tensorflow.keras.layers import Conv1D, MaxPooling1D, Flatten
     10 # Define the CNN model
     11 cnn_model = Sequential([
     12     Conv1D(64, 3, activation='relu', input_shape=
(X_train_cnn.shape[1], X_train_cnn.shape[2])),
     13     MaxPooling1D(2),
   (...)
     20     Dense(4, activation='softmax')  # Assuming 4 classes in 'class'
column
     21 ], name='Improved_CNN_Model')

ModuleNotFoundError: No module named 'tensorflow'
```

```
pip install tensorflow
```

```
Defaulting to user installation because normal site-packages is not writeable
Collecting tensorflow
  Using cached tensorflow-2.16.1-cp311-cp311-win_amd64.whl.metadata (3.5 kB)
Collecting tensorflow-intel==2.16.1 (from tensorflow)
  Using cached tensorflow_intel-2.16.1-cp311-cp311-win_amd64.whl.metadata (5.0 kB)
Collecting absl-py>=1.0.0 (from tensorflow-intel==2.16.1->tensorflow)
  Using cached absl_py-2.1.0-py3-none-any.whl.metadata (2.3 kB)
Collecting astunparse>=1.6.0 (from tensorflow-intel==2.16.1->tensorflow)
  Using cached astunparse-1.6.3-py2.py3-none-any.whl.metadata (4.4 kB)
Collecting flatbuffers>=23.5.26 (from tensorflow-intel==2.16.1->tensorflow)
  Using cached flatbuffers-24.3.25-py2.py3-none-any.whl.metadata (850 bytes)
Collecting gast!=0.5.0,!=0.5.1,!=0.5.2,>=0.2.1 (from tensorflow-intel==2.16.1->tensorflow)
  Using cached gast-0.5.4-py3-none-any.whl.metadata (1.3 kB)
Collecting google-pasta>=0.1.1 (from tensorflow-intel==2.16.1->tensorflow)
  Using cached google_pasta-0.2.0-py3-none-any.whl.metadata (814 bytes)
Collecting h5py>=3.10.0 (from tensorflow-intel==2.16.1->tensorflow)
  Using cached h5py-3.11.0-cp311-cp311-win_amd64.whl.metadata (2.5 kB)
Collecting libclang>=13.0.0 (from tensorflow-intel==2.16.1->tensorflow)
  Using cached libclang-18.1.1-py2.py3-none-win_amd64.whl.metadata (5.3 kB)
Collecting ml-dtypes~=0.3.1 (from tensorflow-intel==2.16.1->tensorflow)
  Using cached ml_dtypes-0.3.2-cp311-cp311-win_amd64.whl.metadata (20 kB)
Collecting opt-einsum>=2.3.2 (from tensorflow-intel==2.16.1->tensorflow)
  Using cached opt_einsum-3.3.0-py3-none-any.whl.metadata (6.5 kB)
Requirement already satisfied: packaging in c:\programdata\anaconda3\lib\site-packages (from tensorflow-intel==2.16.1
Requirement already satisfied: protobuf!=4.21.0,!=4.21.1,!=4.21.2,!=4.21.3,!=4.21.4,!=4.21.5,<5.0.0dev,>=3.20.3 in c
Requirement already satisfied: requests<3,>=2.21.0 in c:\programdata\anaconda3\lib\site-packages (from tensorflow-int
Requirement already satisfied: setuptools in c:\programdata\anaconda3\lib\site-packages (from tensorflow-intel==2.16
Requirement already satisfied: six>=1.12.0 in c:\programdata\anaconda3\lib\site-packages (from tensorflow-intel==2.16
Collecting termcolor>=1.1.0 (from tensorflow-intel==2.16.1->tensorflow)
  Using cached termcolor-2.4.0-py3-none-any.whl.metadata (6.1 kB)
Requirement already satisfied: typing-extensions>=3.6.6 in c:\programdata\anaconda3\lib\site-packages (from tensorflo
Requirement already satisfied: wrapt>=1.11.0 in c:\programdata\anaconda3\lib\site-packages (from tensorflow-intel==2
```

```
Collecting grpcio<2.0,>=1.24.3 (from tensorflow-intel==2.16.1->tensorflow)
  Using cached grpcio-1.62.1-cp311-cp311-win_amd64.whl.metadata (4.2 kB)
Collecting tensorboard<2.17,>=2.16 (from tensorflow-intel==2.16.1->tensorflow)
  Using cached tensorboard-2.16.2-py3-none-any.whl.metadata (1.6 kB)
Collecting keras>=3.0.0 (from tensorflow-intel==2.16.1->tensorflow)
  Using cached keras-3.2.1-py3-none-any.whl.metadata (5.6 kB)
Collecting tensorflow-io-gcs-filesystem>=0.23.1 (from tensorflow-intel==2.16.1->tensorflow)
  Using cached tensorflow_io_gcs_filesystem-0.31.0-cp311-cp311-win_amd64.whl.metadata (14 kB)
Requirement already satisfied: numpy<2.0.0,>=1.23.5 in c:\programdata\anaconda3\lib\site-packages (from tensorflow-in
Requirement already satisfied: wheel<1.0,>=0.23.0 in c:\programdata\anaconda3\lib\site-packages (from astunparse>=1.6
Requirement already satisfied: rich in c:\programdata\anaconda3\lib\site-packages (from keras>=3.0.0->tensorflow-inte
Collecting namex (from keras>=3.0.0->tensorflow-intel==2.16.1->tensorflow)
  Using cached namex-0.0.8-py3-none-any.whl.metadata (246 bytes)
Collecting optree (from keras>=3.0.0->tensorflow-intel==2.16.1->tensorflow)
  Using cached optree-0.11.0-cp311-cp311-win_amd64.whl.metadata (46 kB)
Requirement already satisfied: charset-normalizer<4,>=2 in c:\programdata\anaconda3\lib\site-packages (from requests<
Requirement already satisfied: idna<4,>=2.5 in c:\programdata\anaconda3\lib\site-packages (from requests<3,>=2.21.0->
Requirement already satisfied: urllib3<3,>=1.21.1 in c:\programdata\anaconda3\lib\site-packages (from requests<3,>=2
Requirement already satisfied: certifi>=2017.4.17 in c:\programdata\anaconda3\lib\site-packages (from requests<3,>=2
Requirement already satisfied: markdown>=2.6.8 in c:\programdata\anaconda3\lib\site-packages (from tensorboard<2.17,>
Collecting tensorboard-data-server<0.8.0,>=0.7.0 (from tensorboard<2.17,>=2.16->tensorflow-intel==2.16.1->tensorflow)
  Using cached tensorboard_data_server-0.7.2-py3-none-any.whl.metadata (1.1 kB)
Requirement already satisfied: werkzeug>=1.0.1 in c:\programdata\anaconda3\lib\site-packages (from tensorboard<2.17,>
Requirement already satisfied: MarkupSafe>=2.1.1 in c:\programdata\anaconda3\lib\site-packages (from werkzeug>=1.0.1-
Requirement already satisfied: markdown-it-py<3.0.0,>=2.2.0 in c:\programdata\anaconda3\lib\site-packages (from rich
```

Start coding or generate with AI.

Start coding or generate with AI.

Start coding or generate with AI.

Start coding or generate with AI.

```
X_train.shape
```

```
(240, 8)
```

```
from sklearn.ensemble import RandomForestRegressor
regressor=RandomForestRegressor()
```

```
#A random forest regressor is an estimator that fits a number of classifying decision trees
#on various sub-samples of the dataset and uses averaging to improve the prediction accuracy
#and slos control over-fitting.
```

```
import numpy as np
n_estimators = [int(x) for x in np.linspace(start = 100, stop = 1200, num = 12)]
#n_estimators is a parameter of the random forest regressor which is used to control no of trees in the forest
#so we use 100 200 ....1200 trees for the model
print(n_estimators)
```

```
[100, 200, 300, 400, 500, 600, 700, 800, 900, 1000, 1100, 1200]
```

```
 #Randomized Search CV
# Number of features to consider at every split
max_features = ['auto', 'sqrt'] # we first consider all the featurees and
#then sqare root number of features to train the model

# Maximum number of levels in tree
max_depth = [int(x) for x in np.linspace(5, 30, num = 6)]
#we create trees with 5 10 15 for each model...and train it

# Minimum number of samples required to split a node
min_samples_split = [2, 5, 10, 15, 100]
# we split as 2 nodes forst then 5 then 10 like that till 100 from the list

# Minimum number of samples required at each leaf node
min_samples_leaf = [1, 2, 5, 10]
```

```
max_depth
```

```
[5, 10, 15, 20, 25, 30]
```

```
from sklearn.model_selection import RandomizedSearchCV
#Randomized search on hyper parameters.
#used to select the best parameter for the model


random_grid = {'n_estimators': n_estimators,
               'max_features': max_features,
               'max_depth': max_depth,
               'min_samples_split': min_samples_split,
               'min_samples_leaf': min_samples_leaf}

print(random_grid)
```

```
{'n_estimators': [100, 200, 300, 400, 500, 600, 700, 800, 900, 1000, 1100, 1200], 'max_features': ['auto', 'sqrt'], 'ma
```

```
rf = RandomForestRegressor()
# Random search of parameters, using 3 fold cross validation,
# search across 100 different combinations
rf_random = RandomizedSearchCV(estimator = rf, param_distributions = random_grid,scoring='neg_mean_squared_error', n_iter =


#rf = object of the regressor
#param_distributions = Dictionary with parameters names as keys and parameters as vlaues
#scoring = metric that we used to evaluate the performance of the cross-validated model on the test set.
#we used negative mean squared error.
#cv = 5 fold cross validation
#n_jobs = no of cores to use


rf_random.fit(X_train,y_train)
```

```
Fitting 5 folds for each of 10 candidates, totalling 50 fits
[CV] END max_depth=10, max_features=sqrt, min_samples_leaf=5, min_samples_split=5, n_estimators=900; total time=   2.7s
[CV] END max_depth=10, max_features=sqrt, min_samples_leaf=5, min_samples_split=5, n_estimators=900; total time=   2.6s
[CV] END max_depth=10, max_features=sqrt, min_samples_leaf=5, min_samples_split=5, n_estimators=900; total time=   3.2s
[CV] END max_depth=10, max_features=sqrt, min_samples_leaf=5, min_samples_split=5, n_estimators=900; total time=   2.8s
[CV] END max_depth=10, max_features=sqrt, min_samples_leaf=5, min_samples_split=5, n_estimators=900; total time=   3.0s
[CV] END max_depth=15, max_features=sqrt, min_samples_leaf=2, min_samples_split=10, n_estimators=1100; total time=   4.
[CV] END max_depth=15, max_features=sqrt, min_samples_leaf=2, min_samples_split=10, n_estimators=1100; total time=   3.
[CV] END max_depth=15, max_features=sqrt, min_samples_leaf=2, min_samples_split=10, n_estimators=1100; total time=   5.
[CV] END max_depth=15, max_features=sqrt, min_samples_leaf=2, min_samples_split=10, n_estimators=1100; total time=   3.
[CV] END max_depth=15, max_features=sqrt, min_samples_leaf=2, min_samples_split=10, n_estimators=1100; total time=   2.
C:\ProgramData\anaconda3\Lib\site-packages\sklearn\ensemble\_forest.py:413: FutureWarning: `max_features='auto'` has be
  warn(
[CV] END max_depth=15, max_features=auto, min_samples_leaf=5, min_samples_split=100, n_estimators=300; total time=   0.
C:\ProgramData\anaconda3\Lib\site-packages\sklearn\ensemble\_forest.py:413: FutureWarning: `max_features='auto'` has be
  warn(
[CV] END max_depth=15, max_features=auto, min_samples_leaf=5, min_samples_split=100, n_estimators=300; total time=   0.
C:\ProgramData\anaconda3\Lib\site-packages\sklearn\ensemble\_forest.py:413: FutureWarning: `max_features='auto'` has be
  warn(
[CV] END max_depth=15, max_features=auto, min_samples_leaf=5, min_samples_split=100, n_estimators=300; total time=   0.
C:\ProgramData\anaconda3\Lib\site-packages\sklearn\ensemble\_forest.py:413: FutureWarning: `max_features='auto'` has be
  warn(
[CV] END max_depth=15, max_features=auto, min_samples_leaf=5, min_samples_split=100, n_estimators=300; total time=   0.
C:\ProgramData\anaconda3\Lib\site-packages\sklearn\ensemble\_forest.py:413: FutureWarning: `max_features='auto'` has be
  warn(
[CV] END max_depth=15, max_features=auto, min_samples_leaf=5, min_samples_split=100, n_estimators=300; total time=   1.
C:\ProgramData\anaconda3\Lib\site-packages\sklearn\ensemble\_forest.py:413: FutureWarning: `max_features='auto'` has be
  warn(
[CV] END max_depth=15, max_features=auto, min_samples_leaf=5, min_samples_split=5, n_estimators=400; total time=   1.5s
C:\ProgramData\anaconda3\Lib\site-packages\sklearn\ensemble\_forest.py:413: FutureWarning: `max_features='auto'` has be
  warn(
[CV] END max_depth=15, max_features=auto, min_samples_leaf=5, min_samples_split=5, n_estimators=400; total time=   1.5s
C:\ProgramData\anaconda3\Lib\site-packages\sklearn\ensemble\_forest.py:413: FutureWarning: `max_features='auto'` has be
  warn(
[CV] END max_depth=15, max_features=auto, min_samples_leaf=5, min_samples_split=5, n_estimators=400; total time=   1.1s
C:\ProgramData\anaconda3\Lib\site-packages\sklearn\ensemble\_forest.py:413: FutureWarning: `max_features='auto'` has be
  warn(
[CV] END max_depth=15, max_features=auto, min_samples_leaf=5, min_samples_split=5, n_estimators=400; total time=   1.4s
C:\ProgramData\anaconda3\Lib\site-packages\sklearn\ensemble\_forest.py:413: FutureWarning: `max_features='auto'` has be
  warn(
[CV] END max_depth=15, max_features=auto, min_samples_leaf=5, min_samples_split=5, n_estimators=400; total time=   1.2s
C:\ProgramData\anaconda3\Lib\site-packages\sklearn\ensemble\_forest.py:413: FutureWarning: `max_features='auto'` has be
  warn(
[CV] END max_depth=20, max_features=auto, min_samples_leaf=10, min_samples_split=5, n_estimators=700; total time=   2.1
C:\ProgramData\anaconda3\Lib\site-packages\sklearn\ensemble\_forest.py:413: FutureWarning: `max_features='auto'` has be
  warn(
[CV] END max_depth=20, max_features=auto, min_samples_leaf=10, min_samples_split=5, n_estimators=700; total time=   2.1
C:\ProgramData\anaconda3\Lib\site-packages\sklearn\ensemble\_forest.py:413: FutureWarning: `max_features='auto'` has be
  warn(
[CV] END max_depth=20, max_features=auto, min_samples_leaf=10, min_samples_split=5, n_estimators=700; total time=   2.0
C:\ProgramData\anaconda3\Lib\site-packages\sklearn\ensemble\_forest.py:413: FutureWarning: `max_features='auto'` has be
  warn(
[CV] END max_depth=20, max_features=auto, min_samples_leaf=10, min_samples_split=5, n_estimators=700; total time=   2.2
C:\ProgramData\anaconda3\Lib\site-packages\sklearn\ensemble\_forest.py:413: FutureWarning: `max_features='auto'` has be
  warn(
[CV] END max_depth=20, max_features=auto, min_samples_leaf=10, min_samples_split=5, n_estimators=700; total time=   2.6
[CV] END max_depth=25, max_features=sqrt, min_samples_leaf=1, min_samples_split=2, n_estimators=1000; total time=   3.3
[CV] END max_depth=25, max_features=sqrt, min_samples_leaf=1, min_samples_split=2, n_estimators=1000; total time=   3.3
[CV] END max_depth=25, max_features=sqrt, min_samples_leaf=1, min_samples_split=2, n_estimators=1000; total time=   3.0
[CV] END max_depth=25, max_features=sqrt, min_samples_leaf=1, min_samples_split=2, n_estimators=1000; total time=   3.5
[CV] END max_depth=25, max_features=sqrt, min_samples_leaf=1, min_samples_split=2, n_estimators=1000; total time=   3.8
[CV] END max_depth=5, max_features=sqrt, min_samples_leaf=10, min_samples_split=15, n_estimators=1100; total time=   3.
[CV] END max_depth=5, max_features=sqrt, min_samples_leaf=10, min_samples_split=15, n_estimators=1100; total time=   2.
[CV] END max_depth=5, max_features=sqrt, min_samples_leaf=10, min_samples_split=15, n_estimators=1100; total time=   3.
[CV] END max_depth=5, max_features=sqrt, min_samples_leaf=10, min_samples_split=15, n_estimators=1100; total time=   6.
[CV] END max_depth=5, max_features=sqrt, min_samples_leaf=10, min_samples_split=15, n_estimators=1100; total time=   3.
[CV] END max_depth=15, max_features=sqrt, min_samples_leaf=1, min_samples_split=15, n_estimators=300; total time=   1.1
[CV] END max_depth=15, max_features=sqrt, min_samples_leaf=1, min_samples_split=15, n_estimators=300; total time=   1.1
[CV] END max_depth=15, max_features=sqrt, min_samples_leaf=1, min_samples_split=15, n_estimators=300; total time=   1.0
[CV] END max_depth=15, max_features=sqrt, min_samples_leaf=1, min_samples_split=15, n_estimators=300; total time=   0.8
[CV] END max_depth=15, max_features=sqrt, min_samples_leaf=1, min_samples_split=15, n_estimators=300; total time=   0.8
[CV] END max_depth=5, max_features=sqrt, min_samples_leaf=2, min_samples_split=10, n_estimators=700; total time=   2.2s
[CV] END max_depth=5, max_features=sqrt, min_samples_leaf=2, min_samples_split=10, n_estimators=700; total time=   2.7s
[CV] END max_depth=5, max_features=sqrt, min_samples_leaf=2, min_samples_split=10, n_estimators=700; total time=   2.5s
[CV] END max_depth=5, max_features=sqrt, min_samples_leaf=2, min_samples_split=10, n_estimators=700; total time=   2.1s
[CV] END max_depth=5, max_features=sqrt, min_samples_leaf=2, min_samples_split=10, n_estimators=700; total time=   1.9s
C:\ProgramData\anaconda3\Lib\site-packages\sklearn\ensemble\_forest.py:413: FutureWarning: `max_features='auto'` has be
  warn(
[CV] END max_depth=20, max_features=auto, min_samples_leaf=1, min_samples_split=15, n_estimators=700; total time=   2.3
C:\ProgramData\anaconda3\Lib\site-packages\sklearn\ensemble\_forest.py:413: FutureWarning: `max_features='auto'` has be
  warn(
[CV] END max_depth=20, max_features=auto, min_samples_leaf=1, min_samples_split=15, n_estimators=700; total time=   2.7
```

```
C:\ProgramData\anaconda3\Lib\site-packages\sklearn\ensemble\_forest.py:413: FutureWarning: `max_features='auto'` has be
  warn(
[CV] END max_depth=20, max_features=auto, min_samples_leaf=1, min_samples_split=15, n_estimators=700; total time=   3.8
C:\ProgramData\anaconda3\Lib\site-packages\sklearn\ensemble\_forest.py:413: FutureWarning: `max_features='auto'` has be
  warn(
[CV] END max_depth=20, max_features=auto, min_samples_leaf=1, min_samples_split=15, n_estimators=700; total time=   2.7
C:\ProgramData\anaconda3\Lib\site-packages\sklearn\ensemble\_forest.py:413: FutureWarning: `max_features='auto'` has be
  warn(
[CV] END max_depth=20, max_features=auto, min_samples_leaf=1, min_samples_split=15, n_estimators=700; total time=   2.1
C:\ProgramData\anaconda3\Lib\site-packages\sklearn\ensemble\_forest.py:413: FutureWarning: `max_features='auto'` has be
  warn(
```

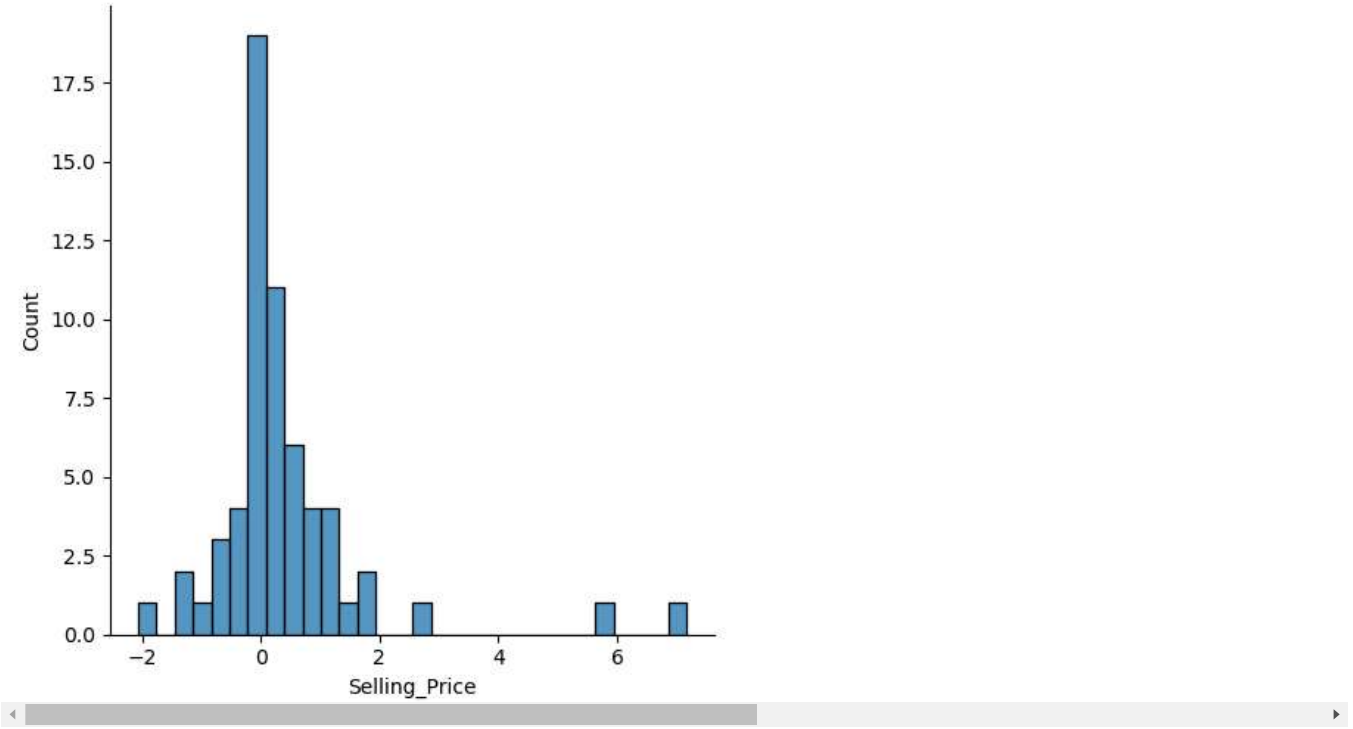
RandomizedSearchCV
estimator: RandomForestRegressor
RandomForestRegressor

```
predictions=rf_random.predict(X_test)
```

```
predictions1=rf_random.predict(X_train)
```

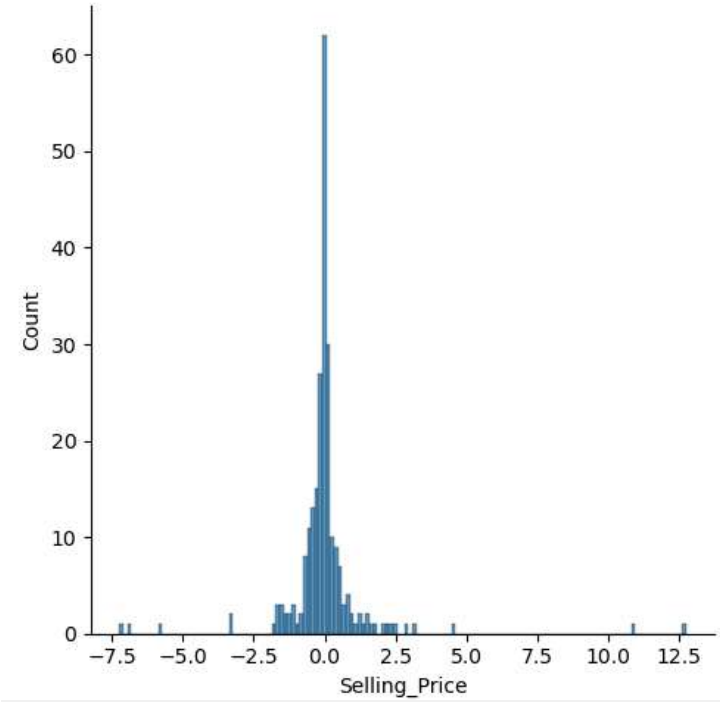
```
sns.displot(y_test-predictions)
```

```
C:\ProgramData\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning: use_inf_as_na option is deprecated
  with pd.option_context('mode.use_inf_as_na', True):
<seaborn.axisgrid.FacetGrid at 0x271735d7d50>
```




```
sns.displot(y_train-predictions1)
```

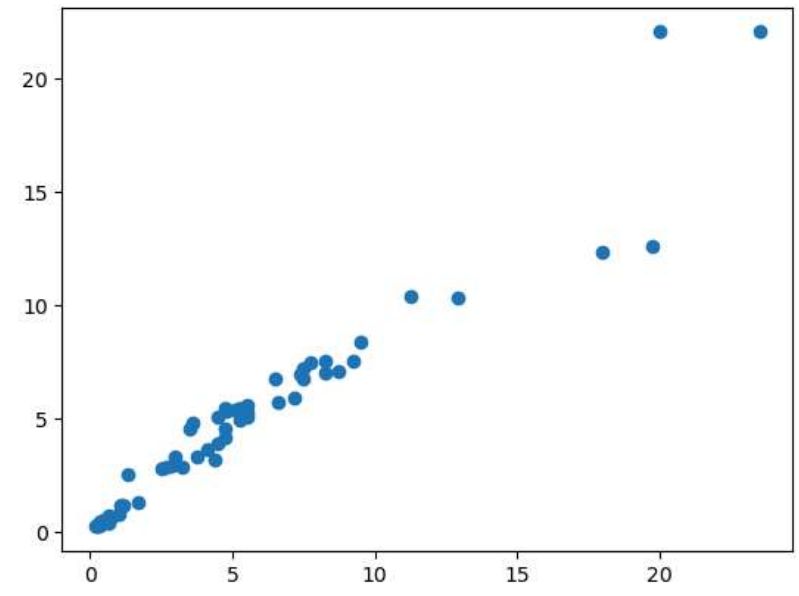
```
C:\ProgramData\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning: use_inf_as_na option is deprecated
  with pd.option_context('mode.use_inf_as_na', True):
<seaborn.axisgrid.FacetGrid at 0x271695d0950>
```




```
#form the plot we can see that almost all the values predicted are correct by comparing the diff between the y_test
#predicted value and we see from the normal distribution of the graph that not much of the samples have varied
#much from the actual value
```

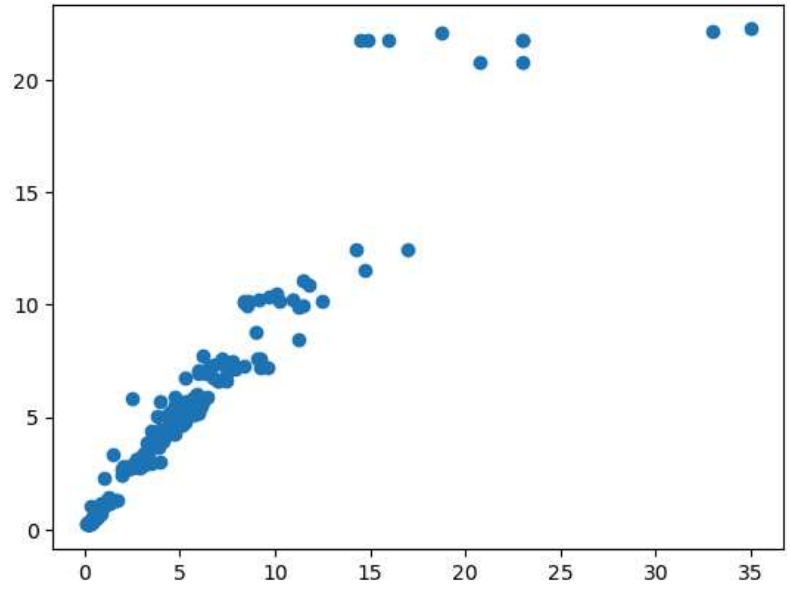
```
plt.scatter(y_test,predictions)
```

```
<matplotlib.collections.PathCollection at 0x271734b7690>
```




```
plt.scatter(y_train,predictions1)
```

```
<matplotlib.collections.PathCollection at 0x271749ff690>
```




```
#even from this points plot we can see that almost all the points are along the
#line y=x which says that the predicted and the y_test values are smae


import pickle
# open a file, where you ant to store the data
file = open('random_forest_regression_model.pkl', 'wb')

# dump information to that file
pickle.dump(rf_random, file)
#we use a pickle file to store the data in a byte stream format


from sklearn.metrics import mean_squared_error
mse = mean_squared_error(y_test,predictions)
rmse = np.sqrt(mse)
print("RMSE : {:.2f}".format(rmse))
```

```
RMSE : 1.39
```

```
from sklearn.metrics import r2_score
r = r2_score(y_test, predictions)
print("R2 score : {}" . format(r))
```

```
R2 score : 0.9233657382908026
```

```
df_check = pd.DataFrame({'Actual': y_test, 'Predicted': predictions})
df_check = df_check.head(25)
#round(df_check,2)
df_check.plot(kind='bar',figsize=(10,5))
plt.grid(which='major', linestyle='-', linewidth='0.5', color='green')
plt.title('Performance of Random Forest')
plt.ylabel('Price')
plt.show()
```