In [1]:
```
import numpy as np
import pandas as pd
```

In [2]:
```
pip install lightgbm
```

```
Requirement already satisfied: lightgbm in c:\users\archa\anaconda3\lib\si
te-packages (4.1.0)
Requirement already satisfied: numpy in c:\users\archa\anaconda3\lib\site-
packages (from lightgbm) (1.23.5)
Requirement already satisfied: scipy in c:\users\archa\anaconda3\lib\site-
packages (from lightgbm) (1.10.0)
Note: you may need to restart the kernel to use updated packages.
```

In [3]:
```
pip install catboost
```

```
Requirement already satisfied: catboost in c:\users\archa\anaconda3\lib\si
te-packages (1.2.2)
Requirement already satisfied: numpy>=1.16.0 in c:\users\archa\anaconda3\l
ib\site-packages (from catboost) (1.23.5)
Requirement already satisfied: plotly in c:\users\archa\anaconda3\lib\site
-packages (from catboost) (5.9.0)
Requirement already satisfied: six in c:\users\archa\anaconda3\lib\site-pa
ckages (from catboost) (1.16.0)
Requirement already satisfied: graphviz in c:\users\archa\anaconda3\lib\si
te-packages (from catboost) (0.20.1)
Requirement already satisfied: matplotlib in c:\users\archa\anaconda3\lib
\site-packages (from catboost) (3.7.0)
Requirement already satisfied: pandas>=0.24 in c:\users\archa\anaconda3\li
b\site-packages (from catboost) (1.5.3)
Requirement already satisfied: scipy in c:\users\archa\anaconda3\lib\site-
packages (from catboost) (1.10.0)
Requirement already satisfied: python-dateutil>=2.8.1 in c:\users\archa\an
aconda3\lib\site-packages (from pandas>=0.24->catboost) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in c:\users\archa\anaconda3\li
b\site-packages (from pandas>=0.24->catboost) (2022.7)
Requirement already satisfied: pillow>=6.2.0 in c:\users\archa\anaconda3\l
ib\site-packages (from matplotlib->catboost) (9.4.0)
Requirement already satisfied: contourpy>=1.0.1 in c:\users\archa\anaconda
3\lib\site-packages (from matplotlib->catboost) (1.0.5)
Requirement already satisfied: pyparsing>=2.3.1 in c:\users\archa\anaconda
3\lib\site-packages (from matplotlib->catboost) (3.0.9)
Requirement already satisfied: fonttools>=4.22.0 in c:\users\archa\anacond
a3\lib\site-packages (from matplotlib->catboost) (4.25.0)
Requirement already satisfied: kiwisolver>=1.0.1 in c:\users\archa\anacond
a3\lib\site-packages (from matplotlib->catboost) (1.4.4)
Requirement already satisfied: packaging>=20.0 in c:\users\archa\anaconda3
\lib\site-packages (from matplotlib->catboost) (22.0)
Requirement already satisfied: cycler>=0.10 in c:\users\archa\anaconda3\li
b\site-packages (from matplotlib->catboost) (0.11.0)
Requirement already satisfied: tenacity>=6.2.0 in c:\users\archa\anaconda3
\lib\site-packages (from plotly->catboost) (8.0.1)
Note: you may need to restart the kernel to use updated packages.
```

In [4]:
```
import numpy as np
import pandas as pd

from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import LabelEncoder
from sklearn.linear_model import LinearRegression, Ridge, Lasso
from sklearn.neighbors import KNeighborsRegressor
from sklearn.neural_network import MLPRegressor
from sklearn.svm import LinearSVR, SVR
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestRegressor, GradientBoostingRegress
from xgboost import XGBRegressor
from lightgbm import LGBMRegressor
from catboost import CatBoostRegressor
import seaborn as sns
from sklearn. linear_model import Lasso

import warnings
warnings.filterwarnings(action='ignore')
```

In [5]:
```
import pandas as pd
data=pd.read_csv('C:/Users/archa/Downloads/dataset Tomato (Price) new.csv')
data
```

Out[5]:

	Date	Incoming Quintel	Lowest Price	Highest Price	Avg_Price	Price	Temperature in CC	Month	Tomat Qualit
0	1/1/2021	106.0	400.0	600.0	500.0	5300.0	17.8	Jan	Goo
1	1/2/2021	130.0	500.0	700.0	600.0	7800.0	18.7	Jan	Goo
2	1/3/2021	152.0	400.0	600.0	400.0	60800.0	19.7	Jan	Goo
3	1/4/2021	104.0	300.0	640.0	475.0	49400.0	20.2	Jan	Goo
4	1/5/2021	95.0	300.0	700.0	400.0	38000.0	20.8	Jan	Averag
...	...	...	...	...	...	...	...	...	.
816	10/2/2023	115.0	500.0	1000.0	750.0	65500.0	29.5	Oct	Goo
817	10/3/2023	126.0	500.0	1000.0	750.0	94500.0	28.0	Oct	Goo
818	10/4/2023	119.0	600.0	900.0	750.0	89200.0	28.0	Oct	Goo
819	10/5/2023	193.0	300.0	900.0	600.0	115400.0	27.0	Oct	Goo
820	10/7/2023	216.0	250.0	750.0	500.0	10800.0	26.0	Oct	Goo

821 rows × 10 columns

In [6]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 821 entries, 0 to 820
Data columns (total 10 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   Date               821 non-null    object 
 1   Incoming Quintel   820 non-null    float64
 2   Lowest Price       820 non-null    float64
 3   Highest Price      820 non-null    float64
 4   Avg_Price          820 non-null    float64
 5   Price              820 non-null    float64
 6   Temperature in CC  821 non-null    float64
 7   Month              821 non-null    object 
 8   Tomato Quality     821 non-null    object 
 9   Season             821 non-null    object 
dtypes: float64(6), object(4)
memory usage: 64.3+ KB
```

In [7]:
```
data.replace({'Season':{'Cloudy':1,'Sunny':2,'Rainy':3}},inplace=True)
data.replace({'Tomato Quality':{'Good': 1,'Bad': 2,'Average': 3}},inplace=T


data.replace({'Month':{
        'Jan': 1,
        'Feb': 2,
        'Mar': 3,
        'Apr': 4,
        'May': 5,
        'Jun': 6,
        'Jul': 7,
        'Aug': 8,
        'Sep': 9,
        'Oct': 10,
        'Nov': 11,
        'Dec': 12}},inplace=True)
data
```

Out[7]:

	Date	Incoming Quintel	Lowest Price	Highest Price	Avg_Price	Price	Temperature in CC	Month	Tomat Qualit
0	1/1/2021	106.0	400.0	600.0	500.0	5300.0	17.8	1	
1	1/2/2021	130.0	500.0	700.0	600.0	7800.0	18.7	1	
2	1/3/2021	152.0	400.0	600.0	400.0	60800.0	19.7	1	
3	1/4/2021	104.0	300.0	640.0	475.0	49400.0	20.2	1	
4	1/5/2021	95.0	300.0	700.0	400.0	38000.0	20.8	1	
...	...	...	...	...	...	...	...	...	.
816	10/2/2023	115.0	500.0	1000.0	750.0	65500.0	29.5	10	
817	10/3/2023	126.0	500.0	1000.0	750.0	94500.0	28.0	10	
818	10/4/2023	119.0	600.0	900.0	750.0	89200.0	28.0	10	
819	10/5/2023	193.0	300.0	900.0	600.0	115400.0	27.0	10	
820	10/7/2023	216.0	250.0	750.0	500.0	10800.0	26.0	10	

821 rows × 10 columns

In [8]:
```
data.replace({'Tomato Quality':{'Good': 1,'Bad': 2,'Average': 3}},inplace=T
data
```

Out[8]:

	Date	Incoming Quintel	Lowest Price	Highest Price	Avg_Price	Price	Temperature in CC	Month	Tomat Qualit
0	1/1/2021	106.0	400.0	600.0	500.0	5300.0	17.8	1	
1	1/2/2021	130.0	500.0	700.0	600.0	7800.0	18.7	1	
2	1/3/2021	152.0	400.0	600.0	400.0	60800.0	19.7	1	
3	1/4/2021	104.0	300.0	640.0	475.0	49400.0	20.2	1	
4	1/5/2021	95.0	300.0	700.0	400.0	38000.0	20.8	1	
...	...	...	...	...	...	...	...	...	.
816	10/2/2023	115.0	500.0	1000.0	750.0	65500.0	29.5	10	
817	10/3/2023	126.0	500.0	1000.0	750.0	94500.0	28.0	10	
818	10/4/2023	119.0	600.0	900.0	750.0	89200.0	28.0	10	
819	10/5/2023	193.0	300.0	900.0	600.0	115400.0	27.0	10	
820	10/7/2023	216.0	250.0	750.0	500.0	10800.0	26.0	10	

821 rows × 10 columns

In [9]:
```
index_to_delete = 236
data = data.drop(index_to_delete)
```

In [10]:
```
data
```

Out[10]:

	Date	Incoming Quintel	Lowest Price	Highest Price	Avg_Price	Price	Temperature in CC	Month	Tomat Qualit
0	1/1/2021	106.0	400.0	600.0	500.0	5300.0	17.8	1	
1	1/2/2021	130.0	500.0	700.0	600.0	7800.0	18.7	1	
2	1/3/2021	152.0	400.0	600.0	400.0	60800.0	19.7	1	
3	1/4/2021	104.0	300.0	640.0	475.0	49400.0	20.2	1	
4	1/5/2021	95.0	300.0	700.0	400.0	38000.0	20.8	1	
...	...	...	...	...	...	...	...	...	.
816	10/2/2023	115.0	500.0	1000.0	750.0	65500.0	29.5	10	
817	10/3/2023	126.0	500.0	1000.0	750.0	94500.0	28.0	10	
818	10/4/2023	119.0	600.0	900.0	750.0	89200.0	28.0	10	
819	10/5/2023	193.0	300.0	900.0	600.0	115400.0	27.0	10	
820	10/7/2023	216.0	250.0	750.0	500.0	10800.0	26.0	10	

820 rows × 10 columns

In [11]:
```
print(data.columns)
data = data.dropna()
```

```
Index(['Date', 'Incoming Quintel', 'Lowest Price', 'Highest Price',
       'Avg_Price', 'Price', 'Temperature in CC', 'Month', 'Tomato Qualit
y',
       'Season'],
      dtype='object')
```

In [12]:
```
import pandas as pd
y = data['Price']
X = data.drop(['Price',"Date"], axis=1)

# #for demand
z = data['Avg_Price']
w = data.drop(['Lowest Price','Highest Price','Date','Incoming Quintel','Av
```

In [13]:
```
y
```

Out[13]:
```
0        5300.0
1        7800.0
2       60800.0
3       49400.0
4       38000.0
         ...
816     65500.0
817     94500.0
818     89200.0
819    115400.0
820     10800.0
Name: Price, Length: 819, dtype: float64
```

In [14]:
```
X
```

Out[14]:

	Incoming Quintel	Lowest Price	Highest Price	Avg_Price	Temperature in CC	Month	Tomato Quality	Season
0	106.0	400.0	600.0	500.0	17.8	1	1	1
1	130.0	500.0	700.0	600.0	18.7	1	1	1
2	152.0	400.0	600.0	400.0	19.7	1	1	1
3	104.0	300.0	640.0	475.0	20.2	1	1	1
4	95.0	300.0	700.0	400.0	20.8	1	3	1
...	...	...	...	...	...	...	...	...
816	115.0	500.0	1000.0	750.0	29.5	10	1	1
817	126.0	500.0	1000.0	750.0	28.0	10	1	1
818	119.0	600.0	900.0	750.0	28.0	10	1	1
819	193.0	300.0	900.0	600.0	27.0	10	1	1
820	216.0	250.0	750.0	500.0	26.0	10	1	1

819 rows × 8 columns

In [15]:
```
print(data.columns)
```

```
Index(['Date', 'Incoming Quintel', 'Lowest Price', 'Highest Price',
       'Avg_Price', 'Price', 'Temperature in CC', 'Month', 'Tomato Qualit
y',
       'Season'],
      dtype='object')
```

In [16]:
```
X_train,X_test,y_train,y_test=train_test_split(X,y,train_size=0.8,shuffle=T
X_train,X_test,y_train,y_test=train_test_split(X,y,train_size=0.8,shuffle=T
w_train,w_test,z_train,z_test=train_test_split(w,z,train_size=0.8,shuffle=T
w_train,w_test,z_train,z_test=train_test_split(w,z,train_size=0.8,shuffle=T
```

In [17]:
```
X_test.shape
```

Out[17]:
```
(164, 8)
```

In [18]:
```
# Check for missing values
print(X_train.isnull().sum())
print(y_train.isnull().sum())

# Check data types
print(X_train.dtypes)
print(y_train.dtypes)
```

```
Incoming Quintel     0
Lowest Price         0
Highest Price        0
Avg_Price            0
Temperature in CC    0
Month                0
Tomato Quality       0
Season               0
dtype: int64
0
Incoming Quintel     float64
Lowest Price         float64
Highest Price        float64
Avg_Price            float64
Temperature in CC    float64
Month                  int64
Tomato Quality         int64
Season                 int64
dtype: object
float64
```

In [19]:
```
np.random.seed(0)
X=np.random.rand(821,10)
y=np.random.rand(821,1)
```

In [20]:
```
print("X_train shape:", X_train.shape)
print("y_train shape:", y_train.shape)
```

```
X_train shape: (655, 8)
y_train shape: (655,)
```

In [21]:
```
print("NaN values in X_train:", X_train.isnull().sum().sum())
print("NaN values in y_train:", y_train.isnull().sum().sum())
```

```
NaN values in X_train: 0
NaN values in y_train: 0
```

In [22]:
```
print("Data types in X_train:", X_train.dtypes.unique())
print("Data type in y_train:", y_train.dtypes)
```

```
Data types in X_train: [dtype('float64') dtype('int64')]
Data type in y_train: float64
```

In [23]:
```
X_train = X_train.values
y_train = y_train.values
```

In [24]:
```
# Create a linear regression model
model = LinearRegression()
# Train the model
model.fit(X_train, y_train)
```

Out[24]:
```
LinearRegression()
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [54]:
```
# model.predict(X_test, y_test)
```

In [25]:
```
models = {
    "                    Linear Regression": LinearRegression(),
    " Linear Regression (L2 Regularization)": Ridge(),
    " Linear Regression (L1 Regularization)": Lasso(),
    "                  K-Nearest Neighbors": KNeighborsRegressor(),
    "                      Neural Network": MLPRegressor(),
    "Support Vector Machine (Linear Kernel)": LinearSVR(),
    "   Support Vector Machine (RBF Kernel)": SVR(),
    "                        Decision Tree": DecisionTreeRegressor(),
    "                        Random Forest": RandomForestRegressor(),
    "                    Gradient Boosting": GradientBoostingRegressor(),
    "                              XGBoost": XGBRegressor(),
    "                             LightGBM": LGBMRegressor(),
    "                             CatBoost": CatBoostRegressor(verbose=0)
}
```

In [26]:
```
for name, model in models.items():
    model.fit(X_train, y_train)
    print(name + " trained.")
for name, model in models.items():
    print(name + " R^2 Score: {:.5f}".format(model.score(X_test, y_test)))
```

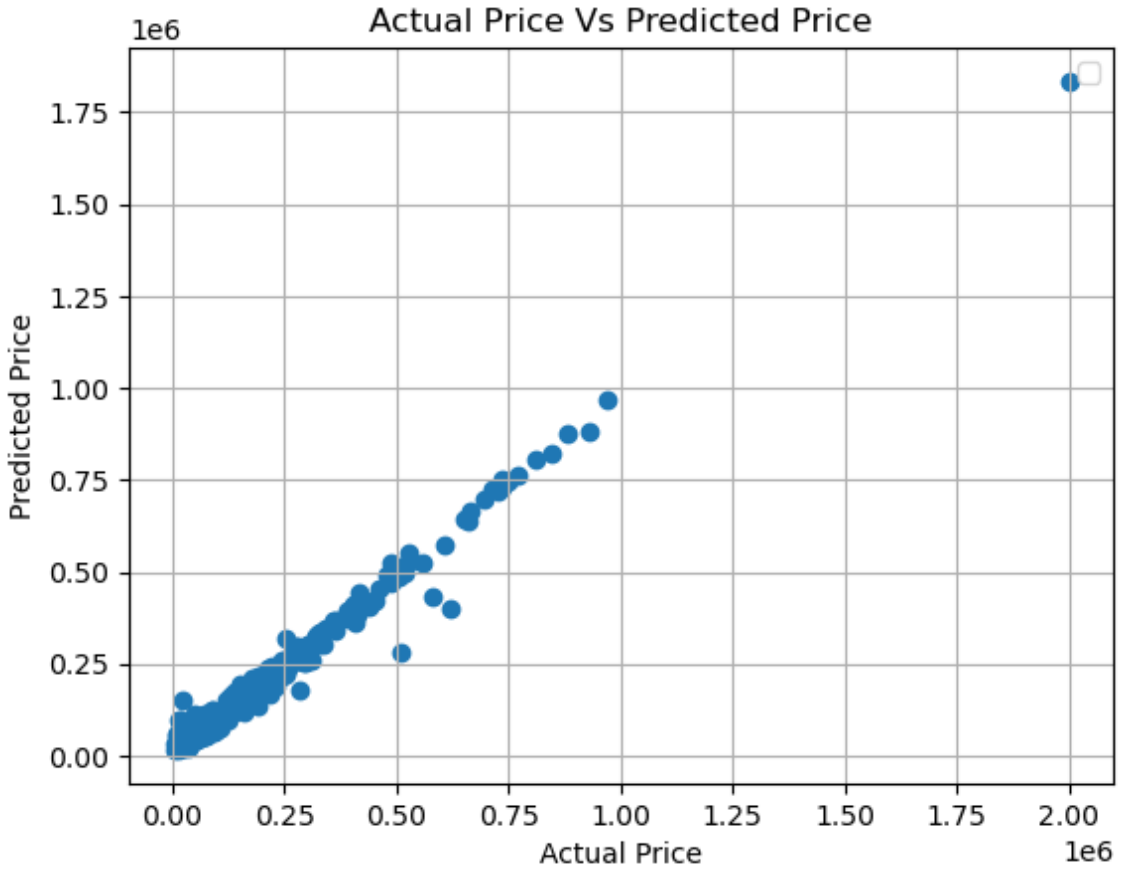
```
                   Linear Regression trained.
 Linear Regression (L2 Regularization) trained.
 Linear Regression (L1 Regularization) trained.
                  K-Nearest Neighbors trained.
                      Neural Network trained.
Support Vector Machine (Linear Kernel) trained.
   Support Vector Machine (RBF Kernel) trained.
                       Decision Tree trained.
                       Random Forest trained.
                   Gradient Boosting trained.
                             XGBoost trained.
[LightGBM] [Info] Auto-choosing col-wise multi-threading, the overhead
of testing was 0.000065 seconds.
You can set `force_col_wise=true` to remove the overhead.
[LightGBM] [Info] Total Bins 346
[LightGBM] [Info] Number of data points in the train set: 655, number o
f used features: 8
[LightGBM] [Info] Start training from score 141367.983206
[LightGBM] [Warning] No further splits with positive gain, best gain: -
```

In [43]:
```
#model selection linear regression
price_reg_model= CatBoostRegressor(verbose=0)
price_reg_model.fit(X_train,y_train)
```

Out[43]:
```
<catboost.core.CatBoostRegressor at 0x12a0d21f820>
```

In [48]:
```
#visualize the actual prices and Predicted prices
from matplotlib import pyplot as plt
#plt.scatter(X_train,y_train, color = 'blue', label = 'Actual')
plt.scatter(y_train,price_data_pred)
plt.xlabel("Actual Price")
plt.ylabel("Predicted Price")
plt.title("Actual Price Vs Predicted Price")
plt.legend()
plt.grid(True)
plt.show()
```

```
No artists with labels found to put in legend.  Note that artists whose la
bel start with an underscore are ignored when legend() is called with no a
rgument.
```



In [46]:
```
data
```

Out[46]:

	Date	Incoming Quintel	Lowest Price	Highest Price	Avg_Price	Price	Temperature in CC	Month	Tomat Qualit
0	1/1/2021	106.0	400.0	600.0	500.0	5300.0	17.8	1	
1	1/2/2021	130.0	500.0	700.0	600.0	7800.0	18.7	1	
2	1/3/2021	152.0	400.0	600.0	400.0	60800.0	19.7	1	
3	1/4/2021	104.0	300.0	640.0	475.0	49400.0	20.2	1	
4	1/5/2021	95.0	300.0	700.0	400.0	38000.0	20.8	1	
...	...	...	...	...	...	...	...	...	.
816	10/2/2023	115.0	500.0	1000.0	750.0	65500.0	29.5	10	
817	10/3/2023	126.0	500.0	1000.0	750.0	94500.0	28.0	10	
818	10/4/2023	119.0	600.0	900.0	750.0	89200.0	28.0	10	
819	10/5/2023	193.0	300.0	900.0	600.0	115400.0	27.0	10	
820	10/7/2023	216.0	250.0	750.0	500.0	10800.0	26.0	10	

819 rows × 10 columns

In [ ]:
```

```