In [1]:

```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import math
from math import sqrt
```

In [3]:

```python
import datetime
import warnings
```

In [8]:

```python
from sklearn.linear_model import LinearRegression
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestRegressor
```

In [10]:

```python
walmart = pd.read_csv('Walmart DataSet.csv')
walmart.head()
```

Out[10]:

	Store	Date	Weekly_Sales	Holiday_Flag	Temperature	Fuel_Price	CPI	Unemployment
0	1	05-02-2010	1643690.90	0	42.31	2.572	211.096358	8.106
1	1	12-02-2010	1641957.44	1	38.51	2.548	211.242170	8.106
2	1	19-02-2010	1611968.17	0	39.93	2.514	211.289143	8.106
3	1	26-02-2010	1409727.59	0	46.63	2.561	211.319643	8.106
4	1	05-03-2010	1554806.68	0	46.50	2.625	211.350143	8.106

In [11]:

```python
walmart.shape
```

Out[11]:

```
(6435, 8)
```

In [12]:

```python
walmart.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6435 entries, 0 to 6434
Data columns (total 8 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   Store         6435 non-null   int64  
 1   Date          6435 non-null   object 
 2   Weekly_Sales  6435 non-null   float64
 3   Holiday_Flag  6435 non-null   int64  
 4   Temperature   6435 non-null   float64
 5   Fuel_Price    6435 non-null   float64
 6   CPI           6435 non-null   float64
 7   Unemployment  6435 non-null   float64
dtypes: float64(5), int64(2), object(1)
memory usage: 402.3+ KB
```

In [13]:

```
walmart["Date"] = pd.to_datetime(walmart["Date"])
walmart['Year'] =walmart['Date'].dt.year
walmart['Month'] =walmart['Date'].dt.month
walmart['Week'] =walmart['Date'].dt.week
```

```
C:\Users\DELL\AppData\Local\Temp\ipykernel_12332\829605467.py:1: UserWarning: Parsing
dates in DD/MM/YYYY format when dayfirst=False (the default) was specified. This may
lead to inconsistently parsed dates! Specify a format to ensure consistent parsing.
  walmart["Date"] = pd.to_datetime(walmart["Date"])
C:\Users\DELL\AppData\Local\Temp\ipykernel_12332\829605467.py:4: FutureWarning: Serie
s.dt.weekofyear and Series.dt.week have been deprecated. Please use Series.dt.isocale
ndar().week instead.
  walmart['Week'] =walmart['Date'].dt.week
```

In [14]:

```
walmart.head()
```

Out[14]:

	Store	Date	Weekly_Sales	Holiday_Flag	Temperature	Fuel_Price	CPI	Unemployment	Ye
0	1	2010-05-02	1643690.90	0	42.31	2.572	211.096358	8.106	20
1	1	2010-12-02	1641957.44	1	38.51	2.548	211.242170	8.106	20
2	1	2010-02-19	1611968.17	0	39.93	2.514	211.289143	8.106	20
3	1	2010-02-26	1409727.59	0	46.63	2.561	211.319643	8.106	20
4	1	2010-05-03	1554806.68	0	46.50	2.625	211.350143	8.106	20

In [15]:

```
walmart.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6435 entries, 0 to 6434
Data columns (total 11 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   Store         6435 non-null   int64
 1   Date          6435 non-null   datetime64[ns]
 2   Weekly_Sales  6435 non-null   float64
 3   Holiday_Flag  6435 non-null   int64
 4   Temperature   6435 non-null   float64
 5   Fuel_Price    6435 non-null   float64
 6   CPI           6435 non-null   float64
 7   Unemployment  6435 non-null   float64
 8   Year          6435 non-null   int64
 9   Month         6435 non-null   int64
 10  Week          6435 non-null   int64
dtypes: datetime64[ns](1), float64(5), int64(5)
memory usage: 553.1 KB
```

In [16]:
```
walmart.describe()
```

Out[16]:

	Store	Weekly_Sales	Holiday_Flag	Temperature	Fuel_Price	CPI	Unemployme
count	6435.000000	6.435000e+03	6435.000000	6435.000000	6435.000000	6435.000000	6435.0000
mean	23.000000	1.046965e+06	0.069930	60.663782	3.358607	171.578394	7.9991
std	12.988182	5.643666e+05	0.255049	18.444933	0.459020	39.356712	1.8758
min	1.000000	2.099862e+05	0.000000	-2.060000	2.472000	126.064000	3.8790
25%	12.000000	5.533501e+05	0.000000	47.460000	2.933000	131.735000	6.8910
50%	23.000000	9.607460e+05	0.000000	62.670000	3.445000	182.616521	7.8740
75%	34.000000	1.420159e+06	0.000000	74.940000	3.735000	212.743293	8.6220
max	45.000000	3.818686e+06	1.000000	100.140000	4.468000	227.232807	14.3130

In [17]:
```
walmart.isnull().sum()
```

Out[17]:
```
Store           0
Date            0
Weekly_Sales    0
Holiday_Flag    0
Temperature     0
Fuel_Price      0
CPI             0
Unemployment    0
Year            0
Month           0
Week            0
dtype: int64
```

In [18]:
```
walmart.duplicated().sum()
```

Out[18]:
```
0
```

In [19]:
```
walmart.groupby('Month')['Weekly_Sales'].mean()
```

Out[19]:

```
Month
1     9.476139e+05
2     1.054597e+06
3     1.024975e+06
4     1.024324e+06
5     1.035379e+06
6     1.064848e+06
7     1.014212e+06
8     1.044874e+06
9     1.009457e+06
10    1.030631e+06
11    1.133751e+06
12    1.210255e+06
Name: Weekly_Sales, dtype: float64
```

In [20]:

```python
walmart.groupby('Year')['Weekly_Sales'].mean()
```

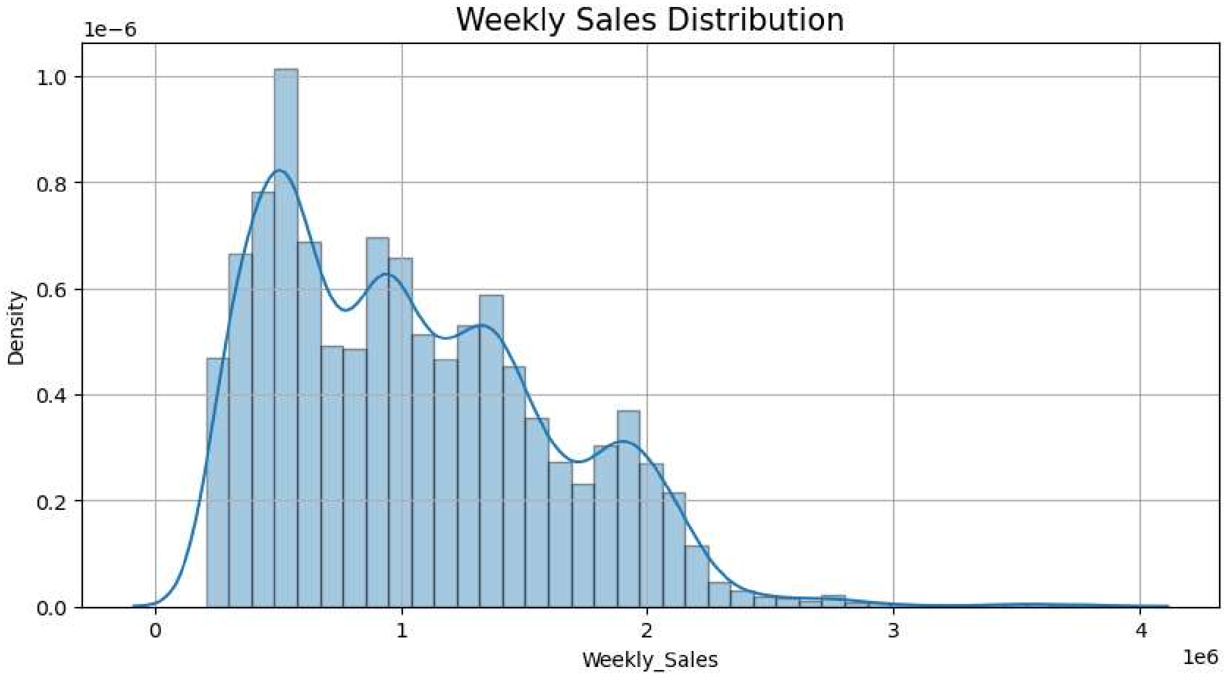
Out[20]:

```
Year
2010    1.059670e+06
2011    1.046239e+06
2012    1.033660e+06
Name: Weekly_Sales, dtype: float64
```

In [21]:

```python
plt.figure(figsize = (10, 5))
sns.distplot(walmart['Weekly_Sales'], hist_kws=dict(edgecolor="black"))
plt.title('Weekly Sales Distribution', fontsize= 15)
plt.grid()
plt.show()
```

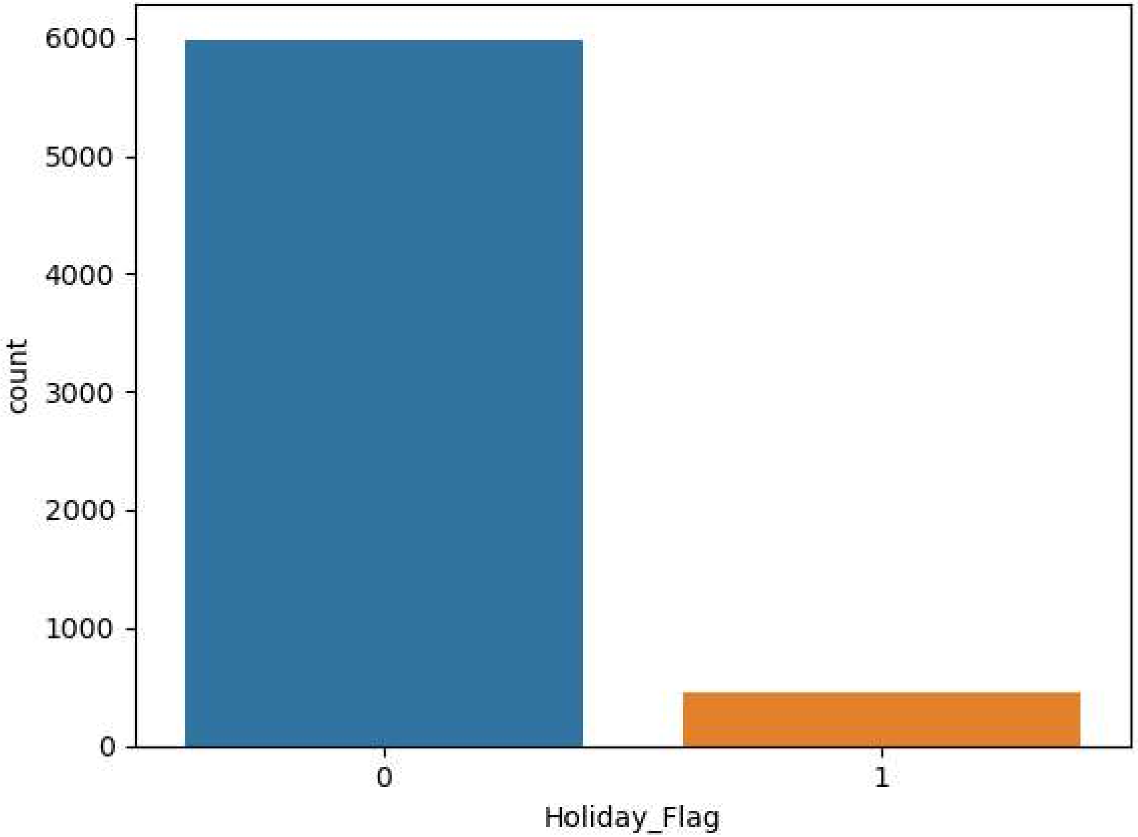
```
C:\Users\DELL\AppData\Local\Temp\ipykernel_12332\2046820045.py:2: UserWarning:

`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(walmart['Weekly_Sales'], hist_kws=dict(edgecolor="black"))
```



In [22]:
```
walmart['Holiday_Flag'].value_counts()
```

Out[22]:
```
0    5985
1     450
Name: Holiday_Flag, dtype: int64
```

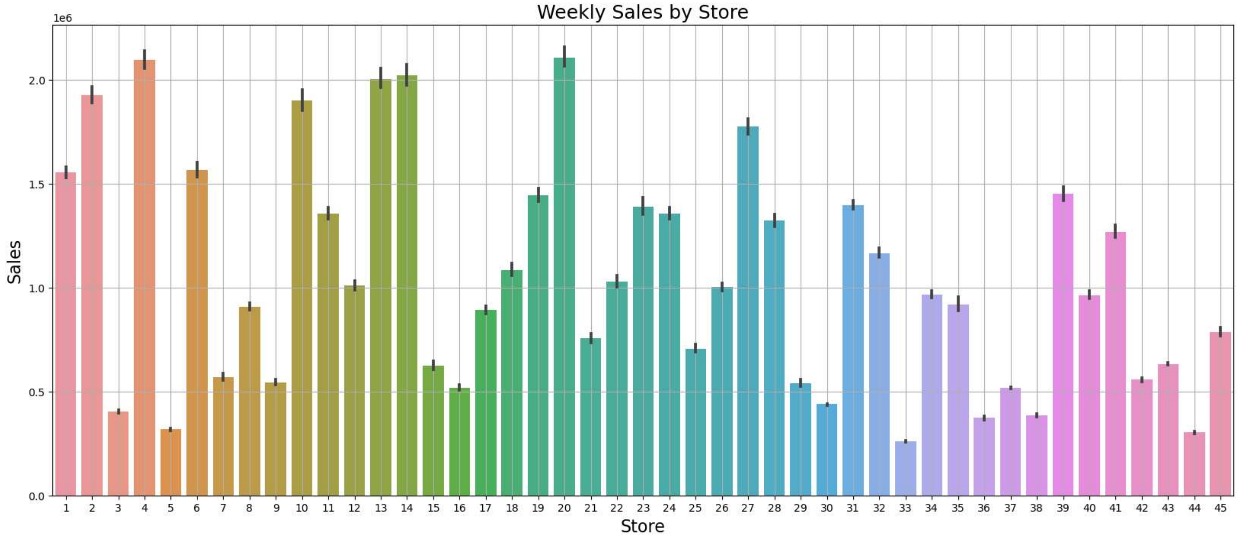
In [23]:
```
sns.countplot(x = 'Holiday_Flag', data = walmart);
```



In [25]:
```
import matplotlib.pyplot as plt
```

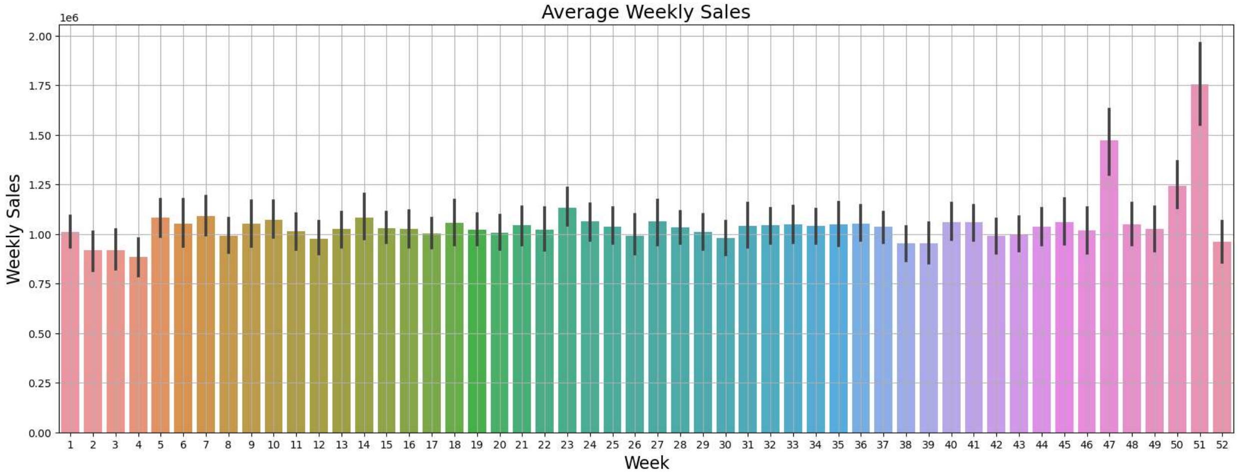
In [29]:

```
plt.figure(figsize=(20,8))
sns.barplot(x=walmart['Store'], y=walmart['Weekly_Sales'])
plt.title('Weekly Sales by Store', fontsize=18)
plt.ylabel('Sales', fontsize=16)
plt.xlabel('Store', fontsize=16)
plt.grid()
plt.show()
```



In [30]:

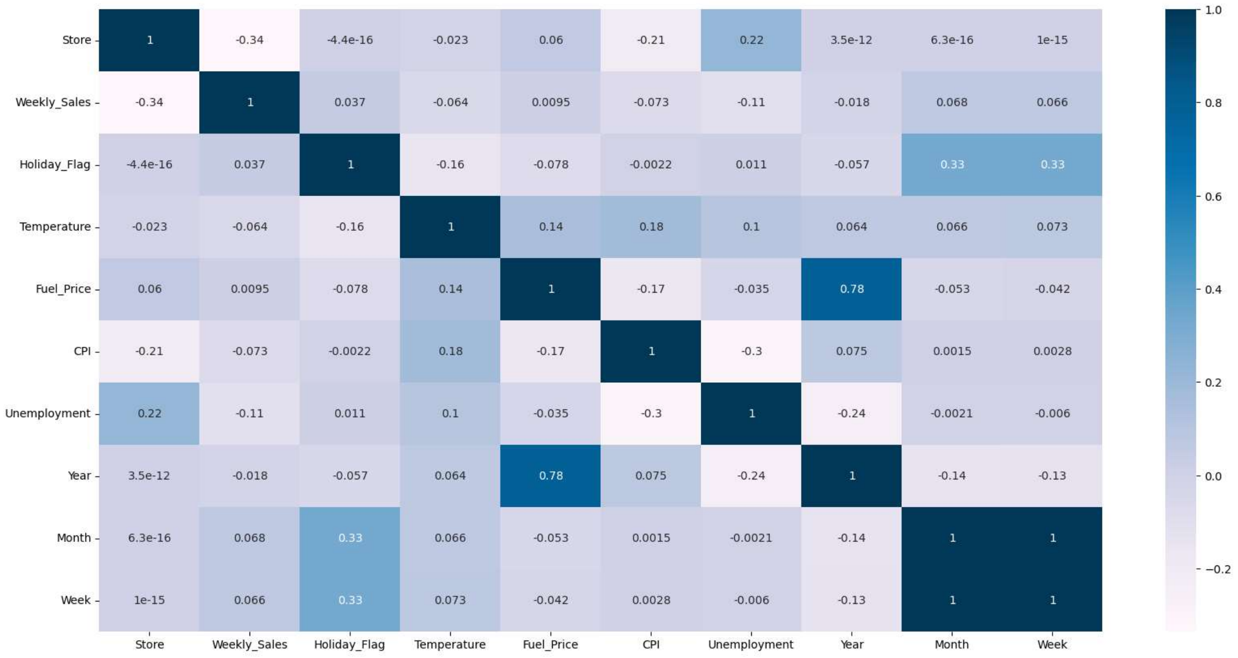
```
plt.figure(figsize = (20, 7))
sns.barplot(x=walmart['Week'], y=walmart['Weekly_Sales'])
plt.title('Average Weekly Sales', fontsize=18)
plt.ylabel('Weekly Sales', fontsize=16)
plt.xlabel('Week', fontsize=16)
plt.grid()
plt.show()
```



In [31]:

```
plt.figure(figsize = (20,10))
sns.heatmap(walmart.corr(), cmap = 'PuBu', annot = True)
plt.show()
```

```
C:\Users\DELL\AppData\Local\Temp\ipykernel_12332\181968183.py:2: FutureWarning: The d
efault value of numeric_only in DataFrame.corr is deprecated. In a future version, it
will default to False. Select only valid columns or specify the value of numeric_only
to silence this warning.
  sns.heatmap(walmart.corr(), cmap = 'PuBu', annot = True)
```



In [32]:
```
walmart.drop(['Temperature', 'Fuel_Price', 'CPI', 'Unemployment'], axis = 1, inplace =
```

In [33]:
```
x = walmart.drop(['Date','Weekly_Sales'], axis=1)
x
```

Out[33]:

	Store	Holiday_Flag	Year	Month	Week
0	1	0	2010	5	17
1	1	1	2010	12	48
2	1	0	2010	2	7
3	1	0	2010	2	8
4	1	0	2010	5	18
...	...	...	...	...	...
6430	45	0	2012	9	39
6431	45	0	2012	5	19
6432	45	0	2012	12	50
6433	45	0	2012	10	42
6434	45	0	2012	10	43

6435 rows × 5 columns

In [34]:
```
y = walmart['Weekly_Sales']
```

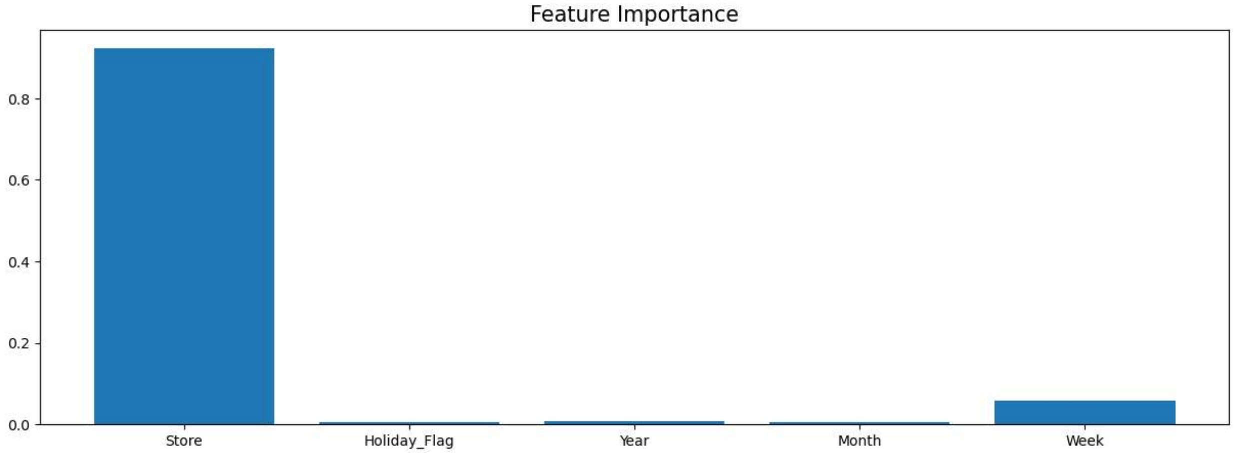
In [35]:
```
rf = RandomForestRegressor(n_estimators = 100)
rf.fit(x, y)
```

Out[35]:

```
▾ RandomForestRegressor
RandomForestRegressor()
```

In [36]:

```
plt.figure(figsize = (15, 5))
plt.bar(x.columns, rf.feature_importances_)
plt.title("Feature Importance", fontsize = 15)
plt.show()
```



In [38]:

```
from sklearn.model_selection import train_test_split
```

In [39]:

```
x_train, x_test, y_train, y_test = train_test_split(x, y, train_size = 0.8, random_sta
```

In [40]:

```
lr = LinearRegression()
lr.fit(x_train, y_train)
```

Out[40]:

```
▾ LinearRegression
LinearRegression()
```

In [41]:

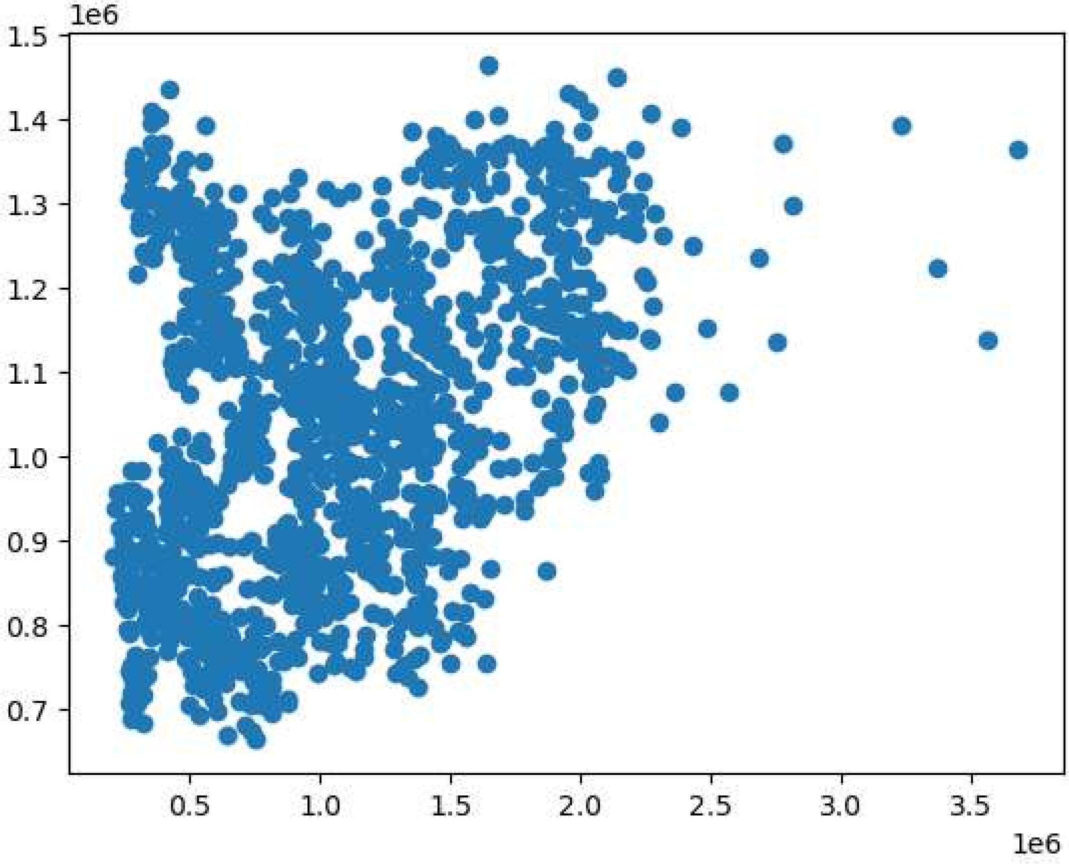
```
y_pred = lr.predict(x_test)
```

In [44]:

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

Out[44]:

```
<matplotlib.collections.PathCollection at 0x210afbc0ee0>
```



In [45]:

```
dtree = DecisionTreeRegressor()
dtree.fit(x_train, y_train)
```

Out[45]:

```
▾ DecisionTreeRegressor
DecisionTreeRegressor()
```

In [46]:

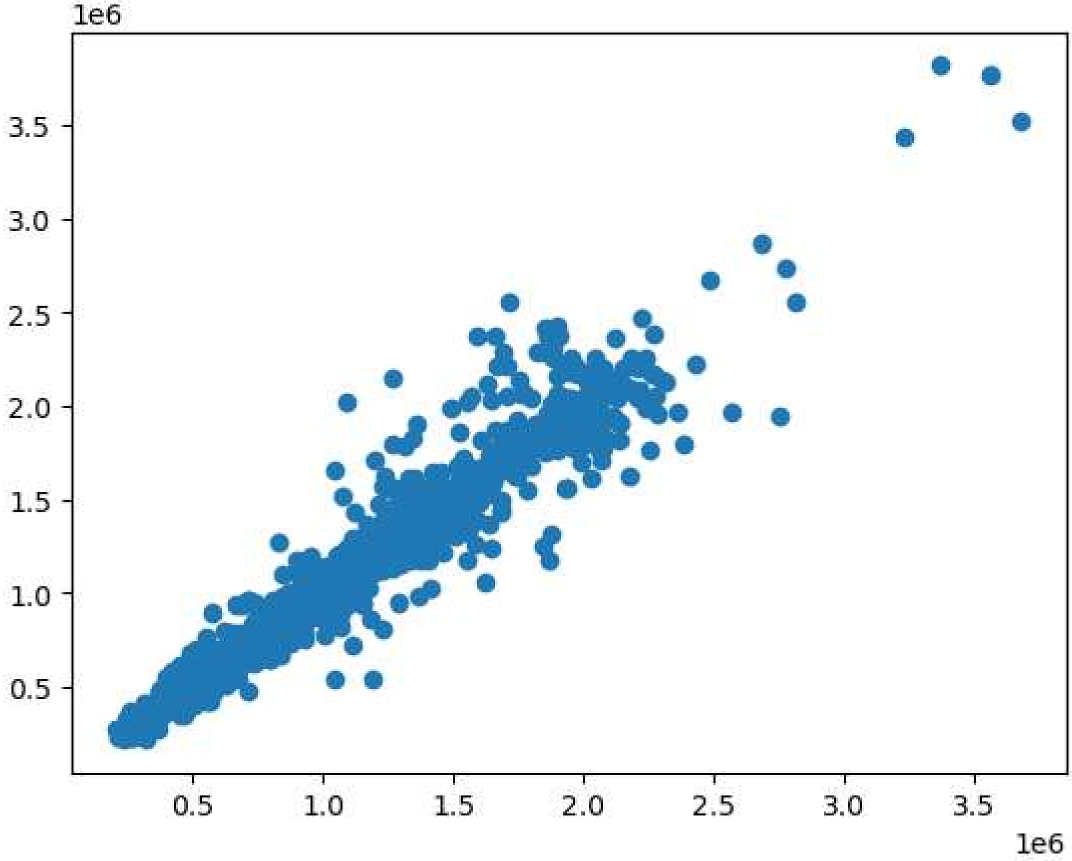
```
y_pred1 = dtree.predict(x_test)
```

In [47]:

```
plt.scatter(y_test, y_pred1)
```

Out[47]:

```
<matplotlib.collections.PathCollection at 0x210afc24a00>
```



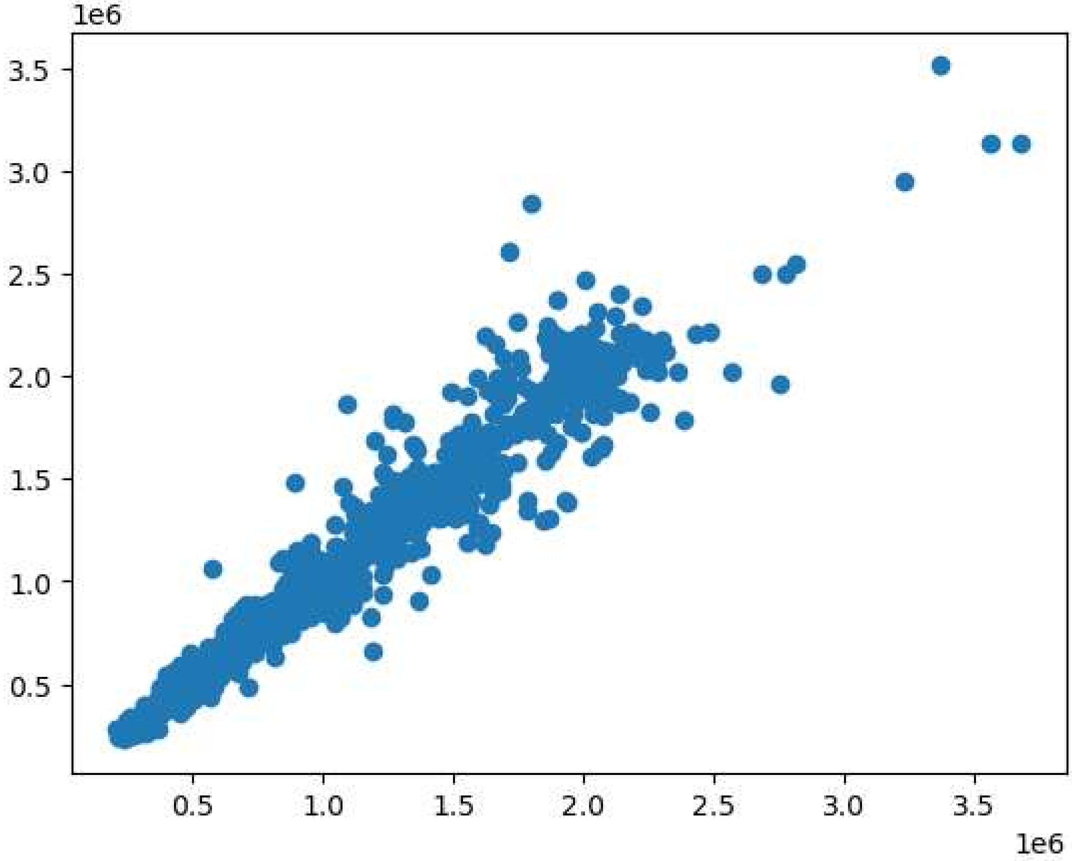
In [48]:
```
rf1 = RandomForestRegressor(n_estimators = 100)
rf1.fit(x_train, y_train)
```

Out[48]:
```
▾ RandomForestRegressor
RandomForestRegressor()
```

In [49]:
```
y_pred2 = rf1.predict(x_test)
```

In [50]:
```
plt.scatter(y_test, y_pred2)
```

Out[50]:
```
<matplotlib.collections.PathCollection at 0x210b3a3eec0>
```



In [51]:

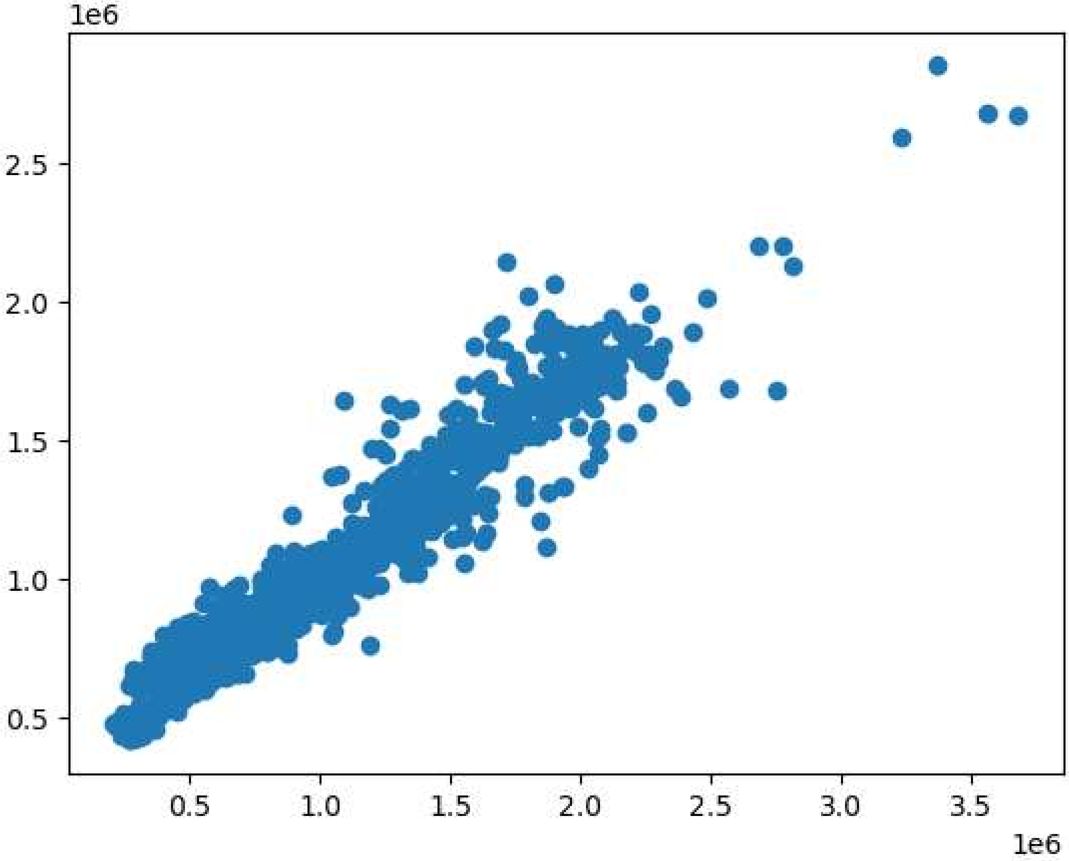
```
y_pred_final = (y_pred + y_pred1  + y_pred2)/3.0
```

In [52]:

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

Out[52]:

```
<matplotlib.collections.PathCollection at 0x210b39ef730>
```



```
In [ ]:
```