In [1]:
```python
# importing necessary liberary
import numpy as np
import pandas as pd
import pymongo
import sklearn
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
import plotly.express as px
import pickle
from scipy import stats
```

In [2]:
```python
import warnings
warnings.filterwarnings('ignore')
```

In [3]:
```python
csv_file_path = 'Walmart.csv'
json_file_path ='Walmart.json'
password='satish321'
connection_string=f'mongodb+srv://satishkandel198:{password}@cluster0.odylp4l.mongodb.net/?retryWrites=true&w=ma
db_name='WALMART'
my_collection='preprocessed_data'
```

In [4]:
```python
# create a function that returns the database connection
def get_connection(connection_string):
    try:
        client=pymongo.MongoClient(connection_string)
        return client
    except Exception as e:
        return  e
```

In [5]:
```python
# importing data from mongo db database into the pandas dataframe
client=get_connection(connection_string)
print(client)

mydb=client[db_name]

collection=mydb[my_collection]
curser=collection.find()

# print(type(curser))
df=pd.DataFrame(curser)
```

```
MongoClient(host=['ac-ny1yb7u-shard-00-01.odylp4l.mongodb.net:27017', 'ac-ny1yb7u-shard-00-02.odylp4l.mongodb.ne
t:27017', 'ac-ny1yb7u-shard-00-00.odylp4l.mongodb.net:27017'], document_class=dict, tz_aware=False, connect=True
, retrywrites=True, w='majority', authsource='admin', replicaset='atlas-vd77c7-shard-0', tls=True)
```

In [6]:
```python
df.drop('_id',axis=1,inplace=True)
```

In [7]:
```python
X = df.drop(['Weekly_Sales'],axis=1)
y = df['Weekly_Sales']
```

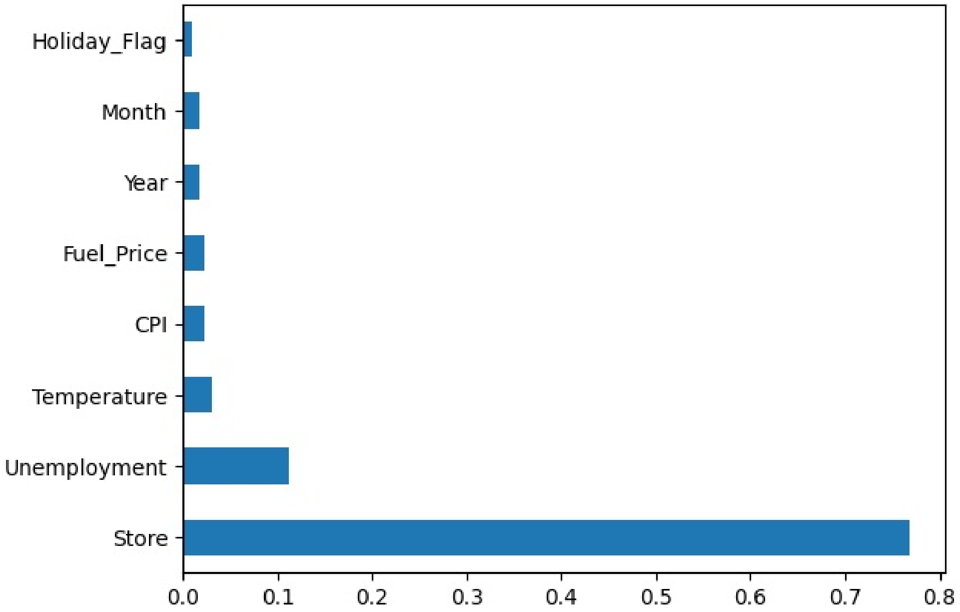
In [8]:
```python
### Feature Importance
from sklearn.ensemble import ExtraTreesRegressor
import matplotlib.pyplot as plt
model = ExtraTreesRegressor()
model.fit(X,y)
```

Out[8]:
```
▾ ExtraTreesRegressor
ExtraTreesRegressor()
```

In [9]:
```python
print(model.feature_importances_)
```

```
[0.76781639 0.00922275 0.03096975 0.021875   0.02261776 0.11214033
 0.         0.01753947 0.01781854]
```

In [10]:
```python
#plot graph of feature importances for better visualization
feat_importances = pd.Series(model.feature_importances_, index=X.columns)
feat_importances.nlargest(8).plot(kind='barh')
plt.show()
```



Based on the feature importance diagram, only few features shows correlation with our target variable 'weekly_sales'. So we are building our prediction model based on top 5 features.

In [11]:
```
cols = ['Temperature', 'Fuel_Price','Unemployment','CPI','Store']
X = df[cols]
```

In [12]:
```
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.3,random_state=42)
```

In [13]:
```
X_train
```

Out[13]:

	Temperature	Fuel_Price	Unemployment	CPI	Store
19	1.322736	-1.549656	0.018746	-1.549656	1
229	0.960343	-0.024970	0.043852	-0.024970	2
4127	0.771275	-1.371684	0.656712	-1.371684	32
668	0.547822	0.552294	-1.639640	0.552294	5
218	1.677391	0.459041	0.043852	0.459041	2
...	...	...	...	...	...
3772	-1.183390	0.343959	0.642560	0.343959	29
5191	-2.058598	-0.410969	-1.756011	-0.410969	40
5226	-0.036025	-1.396596	-0.261484	-1.396596	41
5390	-0.565462	-0.702410	0.649886	-0.702410	42
860	-2.000763	-1.257179	0.714730	-1.257179	7

4143 rows × 5 columns

In [14]:
```
from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import OneHotEncoder
transformer = ColumnTransformer(transformers=[('tf',OneHotEncoder(sparse=False),['Store'])],remainder='passthrou
```

In [15]:
```
X_train= transformer.fit_transform(X_train)
X_test = transformer.transform(X_test)
```

In [16]:
```
X_train[0:10]
```

Out[16]:
```
array([[ 1.        ,  0.        ,  0.        ,  0.        ,  0.        ,
         0.        ,  0.        ,  0.        ,  0.        ,  0.        ,
         0.        ,  0.        ,  0.        ,  0.        ,  0.        ,
         0.        ,  0.        ,  0.        ,  0.        ,  0.        ,
         0.        ,  0.        ,  0.        ,  0.        ,  0.        ,
         0.        ,  0.        ,  0.        ,  0.        ,  0.        ,
         0.        ,  0.        ,  0.        ,  0.        ,  0.        ,
         0.        ,  0.        ,  0.        ,  0.        ,  0.        ,
         0.        ,  0.        ,  0.        ,  0.        ,  0.        ,
         1.32273625, -1.54965628,  0.01874635, -1.54965628],
       [ 0.        ,  1.        ,  0.        ,  0.        ,  0.        ,
         0.        ,  0.        ,  0.        ,  0.        ,  0.        ,
         0.        ,  0.        ,  0.        ,  0.        ,  0.        ,
```

```
  0.        ,  0.        ,  0.        ,  0.        ,  0.        ,
  0.        ,  0.        ,  0.        ,  0.        ,  0.        ,
  0.        ,  0.        ,  0.        ,  0.        ,  0.        ,
  0.        ,  0.        ,  0.        ,  0.        ,  0.        ,
  0.        ,  0.        ,  0.        ,  0.        ,  0.        ,
  0.        ,  0.        ,  0.        ,  0.        ,  0.        ,
  0.96034284, -0.02496988,  0.04385195, -0.02496988],
[ 0.        ,  0.        ,  0.        ,  0.        ,  0.        ,
  0.        ,  0.        ,  0.        ,  0.        ,  0.        ,
  0.        ,  0.        ,  0.        ,  0.        ,  0.        ,
  0.        ,  0.        ,  0.        ,  0.        ,  0.        ,
  0.        ,  0.        ,  0.        ,  0.        ,  0.        ,
  0.        ,  0.        ,  0.        ,  0.        ,  0.        ,
  0.        ,  1.        ,  0.        ,  0.        ,  0.        ,
  0.        ,  0.        ,  0.        ,  0.        ,  0.        ,
  0.        ,  0.        ,  0.        ,  0.        ,  0.        ,
  0.77127491, -1.37168421,  0.65671194, -1.37168421],
[ 0.        ,  0.        ,  0.        ,  0.        ,  1.        ,
  0.        ,  0.        ,  0.        ,  0.        ,  0.        ,
  0.        ,  0.        ,  0.        ,  0.        ,  0.        ,
  0.        ,  0.        ,  0.        ,  0.        ,  0.        ,
  0.        ,  0.        ,  0.        ,  0.        ,  0.        ,
  0.        ,  0.        ,  0.        ,  0.        ,  0.        ,
  0.        ,  0.        ,  0.        ,  0.        ,  0.        ,
  0.        ,  0.        ,  0.        ,  0.        ,  0.        ,
  0.        ,  0.        ,  0.        ,  0.        ,  0.        ,
  0.54782182,  0.5522944 , -1.6396405 ,  0.5522944 ],
[ 0.        ,  1.        ,  0.        ,  0.        ,  0.        ,
  0.        ,  0.        ,  0.        ,  0.        ,  0.        ,
  0.        ,  0.        ,  0.        ,  0.        ,  0.        ,
  0.        ,  0.        ,  0.        ,  0.        ,  0.        ,
  0.        ,  0.        ,  0.        ,  0.        ,  0.        ,
  0.        ,  0.        ,  0.        ,  0.        ,  0.        ,
  0.        ,  0.        ,  0.        ,  0.        ,  0.        ,
  0.        ,  0.        ,  0.        ,  0.        ,  0.        ,
  0.        ,  0.        ,  0.        ,  0.        ,  0.        ,
  1.67739113,  0.45904059,  0.04385195,  0.45904059],
[ 0.        ,  0.        ,  0.        ,  0.        ,  0.        ,
  0.        ,  0.        ,  0.        ,  0.        ,  0.        ,
  0.        ,  0.        ,  0.        ,  0.        ,  0.        ,
  0.        ,  0.        ,  0.        ,  0.        ,  0.        ,
  0.        ,  0.        ,  0.        ,  0.        ,  0.        ,
  0.        ,  0.        ,  0.        ,  0.        ,  0.        ,
  0.        ,  0.        ,  0.        ,  0.        ,  0.        ,
  0.        ,  1.        ,  0.        ,  0.        ,  0.        ,
  0.        ,  0.        ,  0.        ,  0.        ,  0.        ,
  1.48120314, -0.1752032 , -0.47941851, -0.1752032 ],
[ 0.        ,  0.        ,  0.        ,  0.        ,  0.        ,
  0.        ,  0.        ,  0.        ,  0.        ,  0.        ,
  0.        ,  0.        ,  0.        ,  0.        ,  0.        ,
  0.        ,  0.        ,  0.        ,  0.        ,  0.        ,
  0.        ,  0.        ,  0.        ,  0.        ,  0.        ,
  0.        ,  0.        ,  0.        ,  0.        ,  1.        ,
  0.        ,  0.        ,  0.        ,  0.        ,  0.        ,
  0.        ,  0.        ,  0.        ,  0.        ,  0.        ,
  0.        ,  0.        ,  0.        ,  0.        ,  0.        ,
 -0.3712911 ,  0.41472904, -0.43566375,  0.41472904],
[ 0.        ,  0.        ,  0.        ,  0.        ,  0.        ,
  0.        ,  0.        ,  0.        ,  0.        ,  0.        ,
  0.        ,  0.        ,  0.        ,  0.        ,  0.        ,
  1.        ,  0.        ,  0.        ,  0.        ,  0.        ,
  0.        ,  0.        ,  0.        ,  0.        ,  0.        ,
  0.        ,  0.        ,  0.        ,  0.        ,  0.        ,
  0.        ,  0.        ,  0.        ,  0.        ,  0.        ,
  0.        ,  0.        ,  0.        ,  0.        ,  0.        ,
  0.        ,  0.        ,  0.        ,  0.        ,  0.        ,
 -0.68291108,  1.08350591, -1.04687291,  1.08350591],
[ 0.        ,  0.        ,  0.        ,  0.        ,  0.        ,
  0.        ,  0.        ,  0.        ,  0.        ,  0.        ,
  0.        ,  0.        ,  0.        ,  0.        ,  0.        ,
  0.        ,  0.        ,  0.        ,  0.        ,  0.        ,
  0.        ,  0.        ,  0.        ,  0.        ,  0.        ,
  0.        ,  0.        ,  0.        ,  0.        ,  0.        ,
  0.        ,  0.        ,  0.        ,  0.        ,  0.        ,
  0.        ,  0.        ,  0.        ,  0.        ,  0.        ,
  0.        ,  0.        ,  1.        ,  0.        ,  0.        ,
 -0.46567961, -0.75565005,  1.27667983, -0.75565005],
[ 0.        ,  0.        ,  0.        ,  0.        ,  0.        ,
  0.        ,  0.        ,  0.        ,  0.        ,  0.        ,
  0.        ,  0.        ,  0.        ,  0.        ,  0.        ,
  0.        ,  0.        ,  0.        ,  0.        ,  0.        ,
  0.        ,  0.        ,  0.        ,  0.        ,  0.        ,
  0.        ,  0.        ,  0.        ,  0.        ,  0.        ,
```

```
        0.        ,  0.        ,  1.        ,  0.        ,  0.        ,
        0.        ,  0.        ,  0.        ,  0.        ,  0.        ,
        0.        ,  0.        ,  0.        ,  0.        ,  0.        ,
        1.6211201 ,  0.93697939,  0.36608711,  0.93697939]])
```

In [17]:
```
from sklearn.ensemble import RandomForestRegressor
```

In [18]:
```
regressor=RandomForestRegressor()
n_estimators = [int(x) for x in np.linspace(start = 100, stop = 1200, num = 12)]
print(n_estimators)
```

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

In [19]:
```
 #Randomized Search CV

# Number of trees in random forest
n_estimators = [int(x) for x in np.linspace(start = 100, stop = 1200, num = 12)]
# Number of features to consider at every split
max_features = ['auto', 'sqrt']
# Maximum number of levels in tree
max_depth = [int(x) for x in np.linspace(5, 30, num = 6)]
# max_depth.append(None)
# Minimum number of samples required to split a node
min_samples_split = [2, 5, 10, 15, 100]
# Minimum number of samples required at each leaf node
min_samples_leaf = [1, 2, 5, 10]
```

In [20]:
```
# Create the random grid
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
'], 'max_depth': [5, 10, 15, 20, 25, 30], 'min_samples_split': [2, 5, 10, 15, 100], 'min_samples_leaf': [1, 2, 5
, 10]}
```

In [21]:
```
from sklearn.model_selection import RandomizedSearchCV
```

In [22]:
```
# Use the random grid to search for best hyperparameters
# First create the base model to tune
rf = RandomForestRegressor()
```

In [23]:
```
# 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_erro
```

In [24]:
```
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=
9.2s
[CV] END max_depth=10, max_features=sqrt, min_samples_leaf=5, min_samples_split=5, n_estimators=900; total time=
9.5s
[CV] END max_depth=10, max_features=sqrt, min_samples_leaf=5, min_samples_split=5, n_estimators=900; total time=
9.2s
[CV] END max_depth=10, max_features=sqrt, min_samples_leaf=5, min_samples_split=5, n_estimators=900; total time=
10.7s
[CV] END max_depth=10, max_features=sqrt, min_samples_leaf=5, min_samples_split=5, n_estimators=900; total time=
10.1s
[CV] END max_depth=15, max_features=sqrt, min_samples_leaf=2, min_samples_split=10, n_estimators=1100; total tim
e=  15.9s
[CV] END max_depth=15, max_features=sqrt, min_samples_leaf=2, min_samples_split=10, n_estimators=1100; total tim
e=  18.3s
[CV] END max_depth=15, max_features=sqrt, min_samples_leaf=2, min_samples_split=10, n_estimators=1100; total tim
e=  16.1s
[CV] END max_depth=15, max_features=sqrt, min_samples_leaf=2, min_samples_split=10, n_estimators=1100; total tim
e=  16.0s
[CV] END max_depth=15, max_features=sqrt, min_samples_leaf=2, min_samples_split=10, n_estimators=1100; total tim
e=  15.7s
[CV] END max_depth=15, max_features=auto, min_samples_leaf=5, min_samples_split=100, n_estimators=300; total tim
e=   0.0s
[CV] END max_depth=15, max_features=auto, min_samples_leaf=5, min_samples_split=100, n_estimators=300; total tim
e=   0.0s
[CV] END max_depth=15, max_features=auto, min_samples_leaf=5, min_samples_split=100, n_estimators=300; total tim
e=   0.0s
[CV] END max_depth=15, max_features=auto, min_samples_leaf=5, min_samples_split=100, n_estimators=300; total tim
e=   0.0s
[CV] END max_depth=15, max_features=auto, min_samples_leaf=5, min_samples_split=100, n_estimators=300; total tim
e=   0.0s
```

```
[CV] END max_depth=15, max_features=auto, min_samples_leaf=5, min_samples_split=5, n_estimators=400; total time=
0.0s
[CV] END max_depth=15, max_features=auto, min_samples_leaf=5, min_samples_split=5, n_estimators=400; total time=
0.0s
[CV] END max_depth=15, max_features=auto, min_samples_leaf=5, min_samples_split=5, n_estimators=400; total time=
0.0s
[CV] END max_depth=15, max_features=auto, min_samples_leaf=5, min_samples_split=5, n_estimators=400; total time=
0.0s
[CV] END max_depth=15, max_features=auto, min_samples_leaf=5, min_samples_split=5, n_estimators=400; total time=
0.0s
[CV] END max_depth=20, max_features=auto, min_samples_leaf=10, min_samples_split=5, n_estimators=700; total time
=   0.0s
[CV] END max_depth=20, max_features=auto, min_samples_leaf=10, min_samples_split=5, n_estimators=700; total time
=   0.0s
[CV] END max_depth=20, max_features=auto, min_samples_leaf=10, min_samples_split=5, n_estimators=700; total time
=   0.0s
[CV] END max_depth=20, max_features=auto, min_samples_leaf=10, min_samples_split=5, n_estimators=700; total time
=   0.0s
[CV] END max_depth=20, max_features=auto, min_samples_leaf=10, min_samples_split=5, n_estimators=700; total time
=   0.0s
[CV] END max_depth=25, max_features=sqrt, min_samples_leaf=1, min_samples_split=2, n_estimators=1000; total time
=  28.8s
[CV] END max_depth=25, max_features=sqrt, min_samples_leaf=1, min_samples_split=2, n_estimators=1000; total time
=  28.7s
[CV] END max_depth=25, max_features=sqrt, min_samples_leaf=1, min_samples_split=2, n_estimators=1000; total time
=  29.3s
[CV] END max_depth=25, max_features=sqrt, min_samples_leaf=1, min_samples_split=2, n_estimators=1000; total time
=  28.5s
[CV] END max_depth=25, max_features=sqrt, min_samples_leaf=1, min_samples_split=2, n_estimators=1000; total time
=  28.8s
[CV] END max_depth=5, max_features=sqrt, min_samples_leaf=10, min_samples_split=15, n_estimators=1100; total tim
e=   7.0s
[CV] END max_depth=5, max_features=sqrt, min_samples_leaf=10, min_samples_split=15, n_estimators=1100; total tim
e=   7.8s
[CV] END max_depth=5, max_features=sqrt, min_samples_leaf=10, min_samples_split=15, n_estimators=1100; total tim
e=   8.3s
[CV] END max_depth=5, max_features=sqrt, min_samples_leaf=10, min_samples_split=15, n_estimators=1100; total tim
e=   7.4s
[CV] END max_depth=5, max_features=sqrt, min_samples_leaf=10, min_samples_split=15, n_estimators=1100; total tim
e=   8.3s
[CV] END max_depth=15, max_features=sqrt, min_samples_leaf=1, min_samples_split=15, n_estimators=300; total time
=   4.5s
[CV] END max_depth=15, max_features=sqrt, min_samples_leaf=1, min_samples_split=15, n_estimators=300; total time
=   4.3s
[CV] END max_depth=15, max_features=sqrt, min_samples_leaf=1, min_samples_split=15, n_estimators=300; total time
=   4.6s
[CV] END max_depth=15, max_features=sqrt, min_samples_leaf=1, min_samples_split=15, n_estimators=300; total time
=   4.4s
[CV] END max_depth=15, max_features=sqrt, min_samples_leaf=1, min_samples_split=15, n_estimators=300; total time
=   5.1s
[CV] END max_depth=5, max_features=sqrt, min_samples_leaf=2, min_samples_split=10, n_estimators=700; total time=
5.1s
[CV] END max_depth=5, max_features=sqrt, min_samples_leaf=2, min_samples_split=10, n_estimators=700; total time=
5.1s
[CV] END max_depth=5, max_features=sqrt, min_samples_leaf=2, min_samples_split=10, n_estimators=700; total time=
5.0s
[CV] END max_depth=5, max_features=sqrt, min_samples_leaf=2, min_samples_split=10, n_estimators=700; total time=
5.1s
[CV] END max_depth=5, max_features=sqrt, min_samples_leaf=2, min_samples_split=10, n_estimators=700; total time=
5.8s
[CV] END max_depth=20, max_features=auto, min_samples_leaf=1, min_samples_split=15, n_estimators=700; total time
=   0.0s
[CV] END max_depth=20, max_features=auto, min_samples_leaf=1, min_samples_split=15, n_estimators=700; total time
=   0.0s
[CV] END max_depth=20, max_features=auto, min_samples_leaf=1, min_samples_split=15, n_estimators=700; total time
=   0.0s
[CV] END max_depth=20, max_features=auto, min_samples_leaf=1, min_samples_split=15, n_estimators=700; total time
=   0.0s
[CV] END max_depth=20, max_features=auto, min_samples_leaf=1, min_samples_split=15, n_estimators=700; total time
=   0.0s
```

Out[24]:

```
▸ RandomizedSearchCV
▸ estimator: RandomForestRegressor
    ▸ RandomForestRegressor
```

In [25]:

```
rf_random.best_params_
```

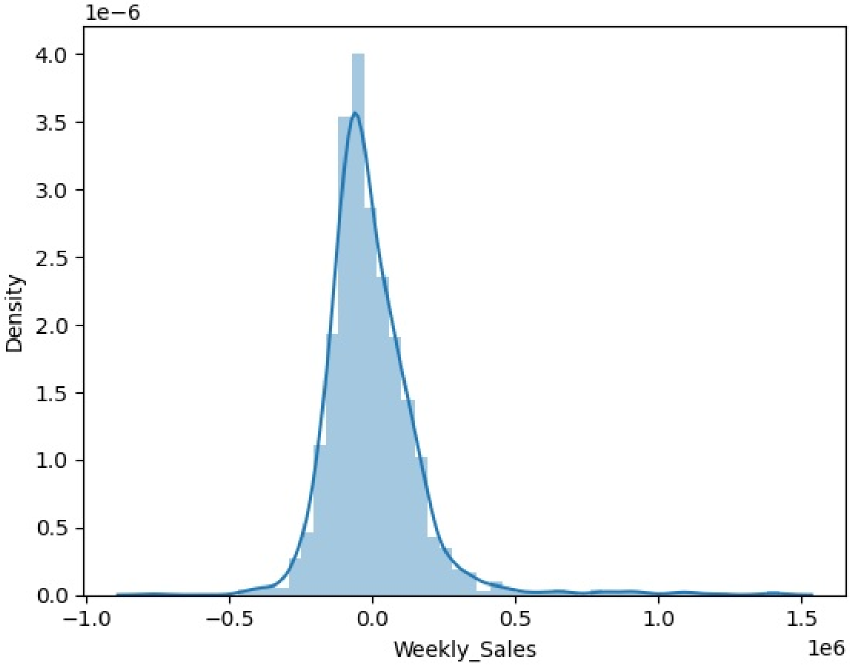
Out[25]:
```
{'n_estimators': 1000,
 'min_samples_split': 2,
 'min_samples_leaf': 1,
 'max_features': 'sqrt',
 'max_depth': 25}
```

In [26]:
```
rf_random.best_score_
```

Out[26]:
```
-33661033192.138744
```

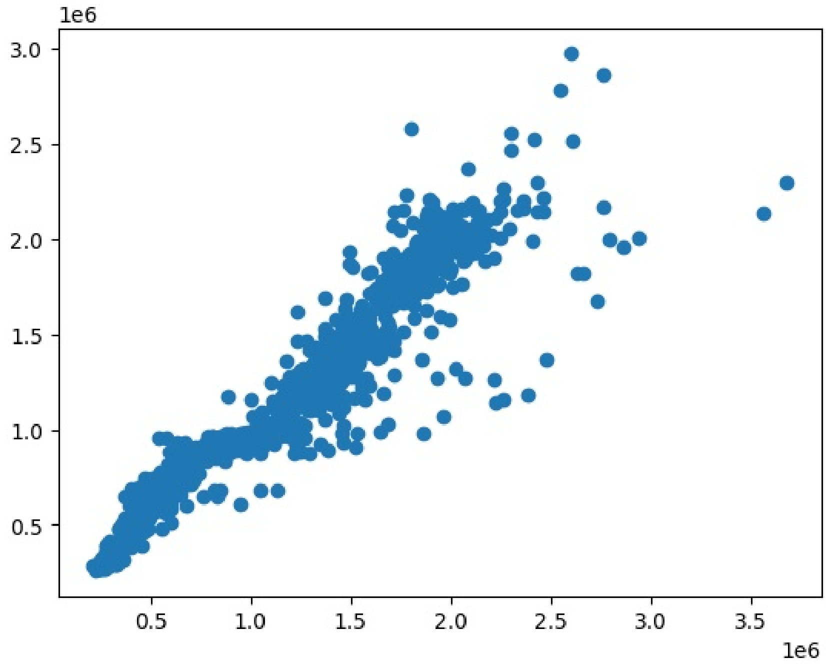
In [27]:
```
predictions=rf_random.predict(X_test)
sns.distplot(y_test-predictions)
```

Out[27]:
```
<Axes: xlabel='Weekly_Sales', ylabel='Density'>
```



In [28]:
```
plt.scatter(y_test,predictions)
```

Out[28]:
```
<matplotlib.collections.PathCollection at 0x14df3898a10>
```



In [29]:
```
# open a file, where you ant to store the data
file = open('random_forest_reg_model.pkl', 'wb')

# dump information to that file
pickle.dump(rf_random, file)
```

Predictive Modelling

Regression Metrics

1. MSE
2. MAE
3. R2 SCORE
4. RMSE
5. Median ABsolute Erro
6. Predictions Error Rate
7. Almost Correct Predictions Error Rate

In [30]:
```
from sklearn.linear_model import LinearRegression,Lasso,Ridge,ElasticNet
from sklearn.metrics import mean_squared_error,mean_absolute_error,r2_score
from sklearn.ensemble import RandomForestRegressor
from sklearn.model_selection import cross_val_score
```

In [31]:
```
def pred_model(model,X_train,Y_train,X_test,Y_test):
    c = model()
    c.fit(X_train,Y_train)
    y_pred = c.predict(X_test)

    print(model)
    print(f'MSE: {mean_squared_error(Y_test,y_pred)}')
    print(f'MAE: {mean_absolute_error(Y_test,y_pred)}')
    print(f'R2 : {r2_score(Y_test,y_pred)}')
```

In [32]:
```
pred_model(LinearRegression,X_train,y_train,X_test,y_test)
```

```
<class 'sklearn.linear_model._base.LinearRegression'>
MSE: 24537082970.468002
MAE: 92687.48701013514
R2 : 0.921969093387086
```

In [33]:
```
pred_model(Lasso,X_train,y_train,X_test,y_test)
```

```
<class 'sklearn.linear_model._coordinate_descent.Lasso'>
MSE: 24519218417.064873
MAE: 92641.42299834719
R2 : 0.9220259048385515
```

In [34]:
```
pred_model(Ridge,X_train,y_train,X_test,y_test)
```

```
<class 'sklearn.linear_model._ridge.Ridge'>
MSE: 24465201092.79901
MAE: 92192.13197227362
R2 : 0.9221976864961487
```

In [35]:
```
pred_model(RandomForestRegressor,X_train,y_train,X_test,y_test)
```

```
<class 'sklearn.ensemble._forest.RandomForestRegressor'>
MSE: 21788372573.71725
MAE: 78718.38447460583
R2 : 0.9307103265863603
```

In [36]:
```
pred_model(ElasticNet,X_train,y_train,X_test,y_test)
```

```
<class 'sklearn.linear_model._coordinate_descent.ElasticNet'>
MSE: 287521684428.2407
MAE: 448797.8752374798
R2 : 0.08564609192500638
```

In [37]:
```
c = LinearRegression()
c.fit(X_train,y_train)
y_pred = c.predict(X_test)
```

In [38]:
```
# open a file, where you ant to store the data
file1 = open('Linear_regression_model.pkl', 'wb')

# dump information to that file
pickle.dump(c, file1)
```

In [39]:
```
# open a file, where you ant to store the data
file2 = open('Lasso_regression_model.pkl', 'wb')

# dump information to that file
pickle.dump(rf_random, file2)
```

In [40]:
```
# open a file, where you ant to store the data
file3 = open('elasticnet_model.pkl', 'wb')

# dump information to that file
pickle.dump(rf_random, file3)
```