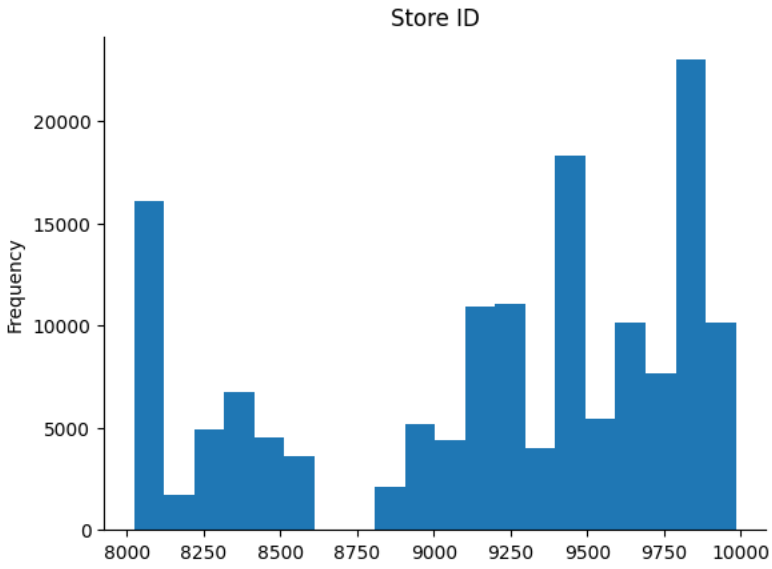
```
#Data Reading
Data = pd.read_csv('/content/drive/MyDrive/PoductDemand.csv')
```

```
from matplotlib import pyplot as plt
Data['Store ID'].plot(kind='hist', bins=20, title='Store ID')
plt.gca().spines[['top', 'right',]].set_visible(False)
```




```
#null values
Data.isnull().sum()
```

```
ID             0
Store ID       0
Total Price    1
Base Price     0
Units Sold     0
dtype: int64
```

```
# Data Distribution
Data
```

	ID	Store ID	Total Price	Base Price	Units Sold
0	1	8091	99.0375	111.8625	20
1	2	8091	99.0375	99.0375	28
2	3	8091	133.9500	133.9500	19
3	4	8091	133.9500	133.9500	44
4	5	8091	141.0750	141.0750	52
...	...	...	...	...	...
150145	212638	9984	235.8375	235.8375	38
150146	212639	9984	235.8375	235.8375	30
150147	212642	9984	357.6750	483.7875	31
150148	212643	9984	141.7875	191.6625	12
150149	212644	9984	234.4125	234.4125	15

150150 rows × 5 columns

```
#importing packages
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error , mean_absolute_error , make_scorer
from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import LabelEncoder
from sklearn.ensemble import RandomForestRegressor, GradientBoostingRegressor
from sklearn.neighbors import KNeighborsRegressor
from sklearn.tree import DecisionTreeRegressor
from sklearn.svm import SVR
```

```
from xgboost import XGBRegressor
from matplotlib import pyplot as plt


#Droping the null value
Data = Data.dropna(axis=0)


def evaluate_regression_models(X, y, test_size=0.2, random_state=42):
    """
    Evaluate multiple regression models on input data.

    Parameters:
    - X: The feature matrix.
    - y: The target variable.
    - test_size: The proportion of data to hold out for testing (default is 0.2).
    - random_state: Seed for random number generation (default is 42).

    Returns:
    - A dictionary containing model names as keys and their mean squared error (MSE) on the test set as values.
    """

    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=test_size, random_state=random_state)

    models = {
        'Linear Regression': LinearRegression(),
        'Random Forest Regressor': RandomForestRegressor(random_state = random_state),
        'Gradient Boosting Regressor': GradientBoostingRegressor(random_state = random_state),
        'K-Nearest Neighbors Regressor': KNeighborsRegressor(),
        'Decision Tree Regressor': DecisionTreeRegressor(random_state = random_state),
        #'Support Vector Regressor': SVR(),
        'XGBoost' : XGBRegressor(random_state = random_state)
    }

    model_scores = pd.DataFrame(columns = ['Model Name' ,'Train MAE' , 'MAE' , 'Train MSE' , 'MSE'])
    modelName = []
    TrainMAE = []
    TrainMSE = []
    MAE = []
    MSE = []

    for model_name, model in models.items():

      model.fit(X_train, y_train)

      y_pred = model.predict(X_test)
      yTest = model.predict(X_train)

      Trainmae = mean_absolute_error(y_train, yTest)

      Trainmse = mean_squared_error(y_train, yTest)

      mae = mean_absolute_error(y_test, y_pred)

      mse = mean_squared_error(y_test, y_pred)

      modelName.append(model_name)
      TrainMAE.append(Trainmae)
      TrainMSE.append(Trainmse)
      MAE.append(mae)
      MSE.append(mse)

      print(f'{model_name}\nMean Absolute Error :\n Train : {mae}\n Test : {Trainmae}\nMean Squared Error :\n Train : {mse}\n Test : {Tra


    model_scores['Model Name'] = modelName
    model_scores['MAE'] = MAE
    model_scores['MSE'] = MSE
    model_scores['Train MAE'] = TrainMAE
    model_scores['Train MSE'] = TrainMSE
    return model_scores , models


#encoding storeid
storeid = Data['Store ID']
EncodedVal = LabelEncoder().fit_transform(storeid)
EncodedVal

     array([ 3,  3,  3, ..., 75, 75, 75])


#replacing the original with encoded values
Data [ 'Store ID'] = EncodedVal
```

```
<ipython-input-9-3626f344f05e>:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus
  Data [ 'Store ID'] = EncodedVal
```

```
# Seprating X and Y values
X = Data.drop(['ID','Units Sold'],axis=1)
Y = Data['Units Sold']


# Finding the best model
score,models = evaluate_regression_models(X,Y)
```

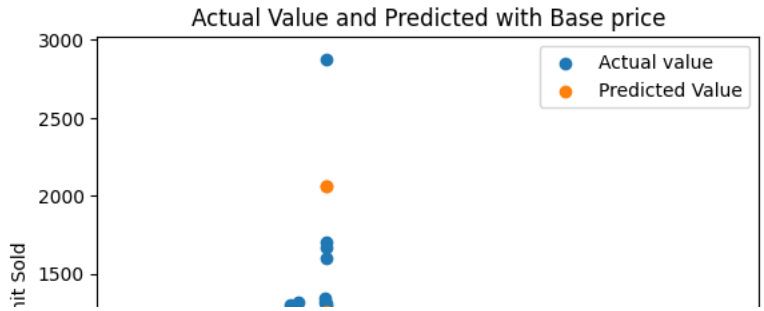
```
Linear Regression
Mean Absolute Error :
 Train : 32.48619926013835
 Test : 32.971777555407584
Mean Squared Error :
 Train : 2785.4886067716207
 Test : 3168.0524425033855
Random Forest Regressor
Mean Absolute Error :
 Train : 18.92889385404334
 Test : 13.481957943240586
Mean Squared Error :
 Train : 1268.4901257638048
 Test : 549.882584513318
Gradient Boosting Regressor
Mean Absolute Error :
 Train : 25.559876376161668
 Test : 25.93189620584567
Mean Squared Error :
 Train : 1873.9426544306266
 Test : 2141.437465698987
K-Nearest Neighbors Regressor
Mean Absolute Error :
 Train : 21.256190476190476
 Test : 18.55506123094598
Mean Squared Error :
 Train : 1612.4435391275392
 Test : 1271.6142881642372
Decision Tree Regressor
Mean Absolute Error :
 Train : 20.556651344992627
 Test : 11.07446679416054
Mean Squared Error :
 Train : 1777.489385179284
 Test : 416.0701821379953
XGBoost
Mean Absolute Error :
 Train : 20.094529283048832
 Test : 19.563251202500055
Mean Squared Error :
 Train : 1291.6874922881454
 Test : 1184.4253677549048
```

```
# so the best model is Random Forest Regressor


# prediction using the model
pred = models['Random Forest Regressor'].predict(X)


plt.title('Actual Value and Predicted with Base price')
plt.scatter(X['Base Price'],Y)
plt.scatter(X['Base Price'],pred)
plt.legend(['Actual value','Predicted Value'])
plt.xlabel('Base Price')
plt.ylabel('Unit Sold')
```

```
Text(0, 0.5, 'Unit Sold')
```




```
plt.title('Actual Value and Predicted with Total price')
plt.scatter(X['Total Price'],Y)
plt.scatter(X['Total Price'],pred)
plt.legend(['Actual value','Predicted Value'])
plt.xlabel('Total Price')
plt.ylabel('Unit Sold')
```

```
Text(0, 0.5, 'Unit Sold')
```