In [4]:
```
import pandas as pd
data = pd.read_csv(r"C:\Users\manju\Desktop\HousingData.csv")
data
df=pd.DataFrame(data)
df
```

Out[4]:

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	L
0	0.00632	18.0	2.31	0.0	0.538	6.575	65.2	4.0900	1	296	15.3	396.90	
1	0.02731	0.0	7.07	0.0	0.469	6.421	78.9	4.9671	2	242	17.8	396.90	
2	0.02729	0.0	7.07	0.0	0.469	7.185	61.1	4.9671	2	242	17.8	392.83	
3	0.03237	0.0	2.18	0.0	0.458	6.998	45.8	6.0622	3	222	18.7	394.63	
4	0.06905	0.0	2.18	0.0	0.458	7.147	54.2	6.0622	3	222	18.7	396.90	
...	...	...	...	...	...	...	...	...	...	...	...	...	
501	0.06263	0.0	11.93	0.0	0.573	6.593	69.1	2.4786	1	273	21.0	391.99	
502	0.04527	0.0	11.93	0.0	0.573	6.120	76.7	2.2875	1	273	21.0	396.90	
503	0.06076	0.0	11.93	0.0	0.573	6.976	91.0	2.1675	1	273	21.0	396.90	
504	0.10959	0.0	11.93	0.0	0.573	6.794	89.3	2.3889	1	273	21.0	393.45	
505	0.04741	0.0	11.93	0.0	0.573	6.030	NaN	2.5050	1	273	21.0	396.90	

506 rows × 14 columns

Boston house data set

Data spliting

In [4]:
```
import pandas as pd
data = pd.read_csv(r"C:\Users\manju\Desktop\HousingData.csv")

```

In [6]:

```
X = data.drop('AGE', axis=1)
y = data['ZN']
from sklearn.model_selection import train_test_split
X_train, X_temp, y_train, y_temp = train_test_split(X, y, test_size=0.3
X_val, X_test, y_val, y_test = train_test_split(X_temp, y_temp, test_si
print(X)
print(y)

```

```
        CRIM    ZN  INDUS  CHAS    NOX     RM     DIS  RAD  TAX  PTRATIO  \
0    0.00632  18.0   2.31   0.0  0.538  6.575  4.0900    1  296     15.3
1    0.02731   0.0   7.07   0.0  0.469  6.421  4.9671    2  242     17.8
2    0.02729   0.0   7.07   0.0  0.469  7.185  4.9671    2  242     17.8
3    0.03237   0.0   2.18   0.0  0.458  6.998  6.0622    3  222     18.7
4    0.06905   0.0   2.18   0.0  0.458  7.147  6.0622    3  222     18.7
..       ...   ...    ...   ...    ...    ...     ...  ...  ...      ...
501  0.06263   0.0  11.93   0.0  0.573  6.593  2.4786    1  273     21.0
502  0.04527   0.0  11.93   0.0  0.573  6.120  2.2875    1  273     21.0
503  0.06076   0.0  11.93   0.0  0.573  6.976  2.1675    1  273     21.0
504  0.10959   0.0  11.93   0.0  0.573  6.794  2.3889    1  273     21.0
505  0.04741   0.0  11.93   0.0  0.573  6.030  2.5050    1  273     21.0

          B  LSTAT  MEDV
0    396.90   4.98  24.0
1    396.90   9.14  21.6
2    392.83   4.03  34.7
3    394.63   2.94  33.4
4    396.90    NaN  36.2
..      ...    ...   ...
501  391.99    NaN  22.4
502  396.90   9.08  20.6
503  396.90   5.64  23.9
504  393.45   6.48  22.0
505  396.90   7.88  11.9

[506 rows x 13 columns]
0      18.0
1       0.0
2       0.0
3       0.0
4       0.0
       ...
501     0.0
502     0.0
503     0.0
504     0.0
505     0.0
Name: ZN, Length: 506, dtype: float64
```

train dataset

In [7]:

```
import pandas as pd
from sklearn.model_selection import train_test_split
data = pd.read_csv(r"C:\Users\manju\Desktop\HousingData.csv")
X = data.drop('AGE', axis=1)
y = data['TAX']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2
X
```

Out[7]:

	CRIM	ZN	INDUS	CHAS	NOX	RM	DIS	RAD	TAX	PTRATIO	B	LSTAT
0	0.00632	18.0	2.31	0.0	0.538	6.575	4.0900	1	296	15.3	396.90	4.98
1	0.02731	0.0	7.07	0.0	0.469	6.421	4.9671	2	242	17.8	396.90	9.14
2	0.02729	0.0	7.07	0.0	0.469	7.185	4.9671	2	242	17.8	392.83	4.03
3	0.03237	0.0	2.18	0.0	0.458	6.998	6.0622	3	222	18.7	394.63	2.94
4	0.06905	0.0	2.18	0.0	0.458	7.147	6.0622	3	222	18.7	396.90	NaN
...	...	...	...	...	...	...	...	...	...	...	...	...
501	0.06263	0.0	11.93	0.0	0.573	6.593	2.4786	1	273	21.0	391.99	NaN
502	0.04527	0.0	11.93	0.0	0.573	6.120	2.2875	1	273	21.0	396.90	9.08
503	0.06076	0.0	11.93	0.0	0.573	6.976	2.1675	1	273	21.0	396.90	5.64
504	0.10959	0.0	11.93	0.0	0.573	6.794	2.3889	1	273	21.0	393.45	6.48
505	0.04741	0.0	11.93	0.0	0.573	6.030	2.5050	1	273	21.0	396.90	7.88

506 rows × 13 columns

Validation dataset

In [11]:

```
import csv
from sklearn.model_selection import train_test_split
data = []
with open(r"C:\Users\manju\Desktop\HousingData.csv")as file:
    csv_reader = csv.reader(file)
    for row in csv_reader:
        data.append(row)
X = [row[:-1] for row in data]
y = [row[-1] for row in data]
X_train, X_val, y_train, y_val = train_test_split(X, y, test_size=0.2, r
X
```

Out[11]:

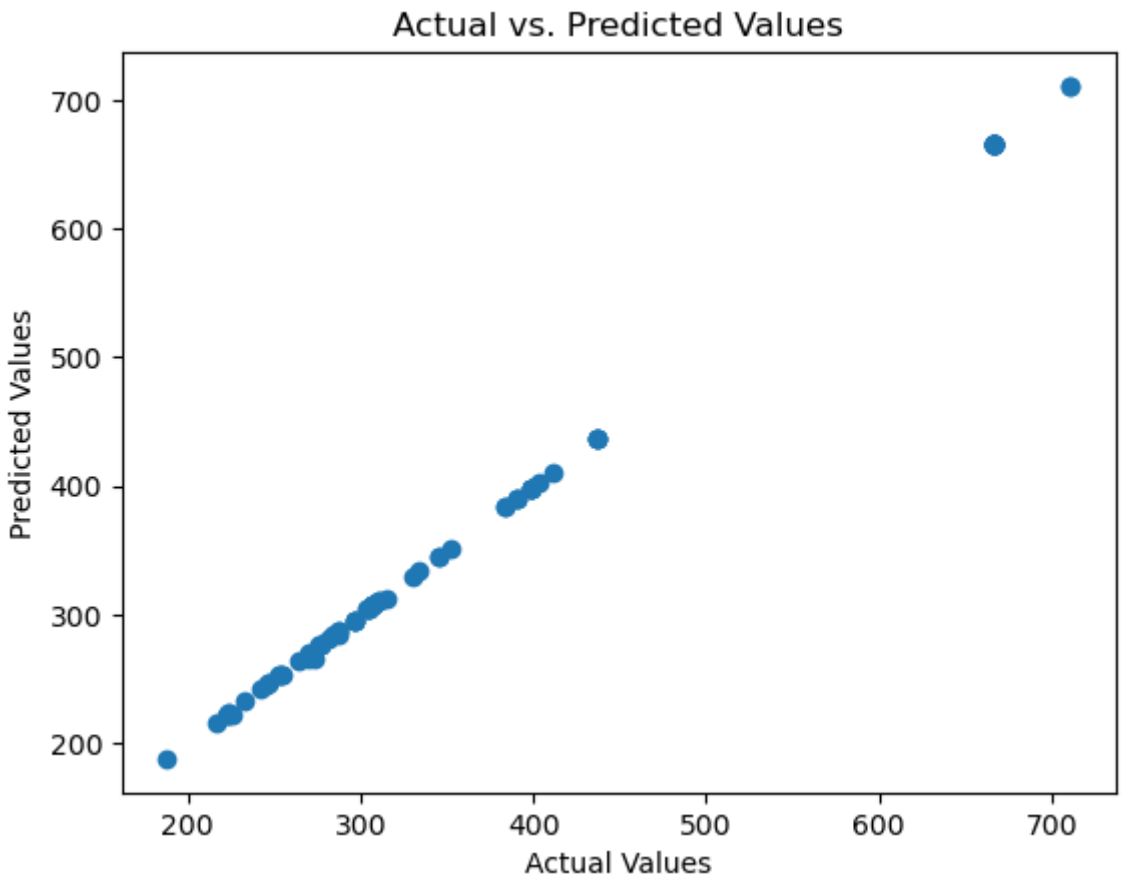
```
[['CRIM',
  'ZN',
  'INDUS',
  'CHAS',
  'NOX',
  'RM',
  'AGE',
  'DIS',
  'RAD',
  'TAX',
  'PTRATIO',
  'B',
  'LSTAT'],
 ['0.00632',
  '18',
  '2.31',
  '0',
  '0.538',
  '6.575',
```

Overfitting

In [14]:

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeRegressor
from sklearn.metrics import mean_squared_error
import matplotlib.pyplot as plt
data = pd.read_csv(r"C:\Users\manju\Desktop\HousingData.csv")
X = data.drop('AGE', axis=1)
y = data['TAX']
X.fillna(X.mean(), inplace=True)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
model = DecisionTreeRegressor(max_depth=None)
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
mse = mean_squared_error(y_test, y_pred)
print(f"Mean Squared Error: {mse}")
plt.scatter(y_test, y_pred)
plt.xlabel("Actual Values")
plt.ylabel("Predicted Values")
plt.title("Actual vs. Predicted Values")
plt.show()

```

```
Mean Squared Error: 1.2843137254901962
```

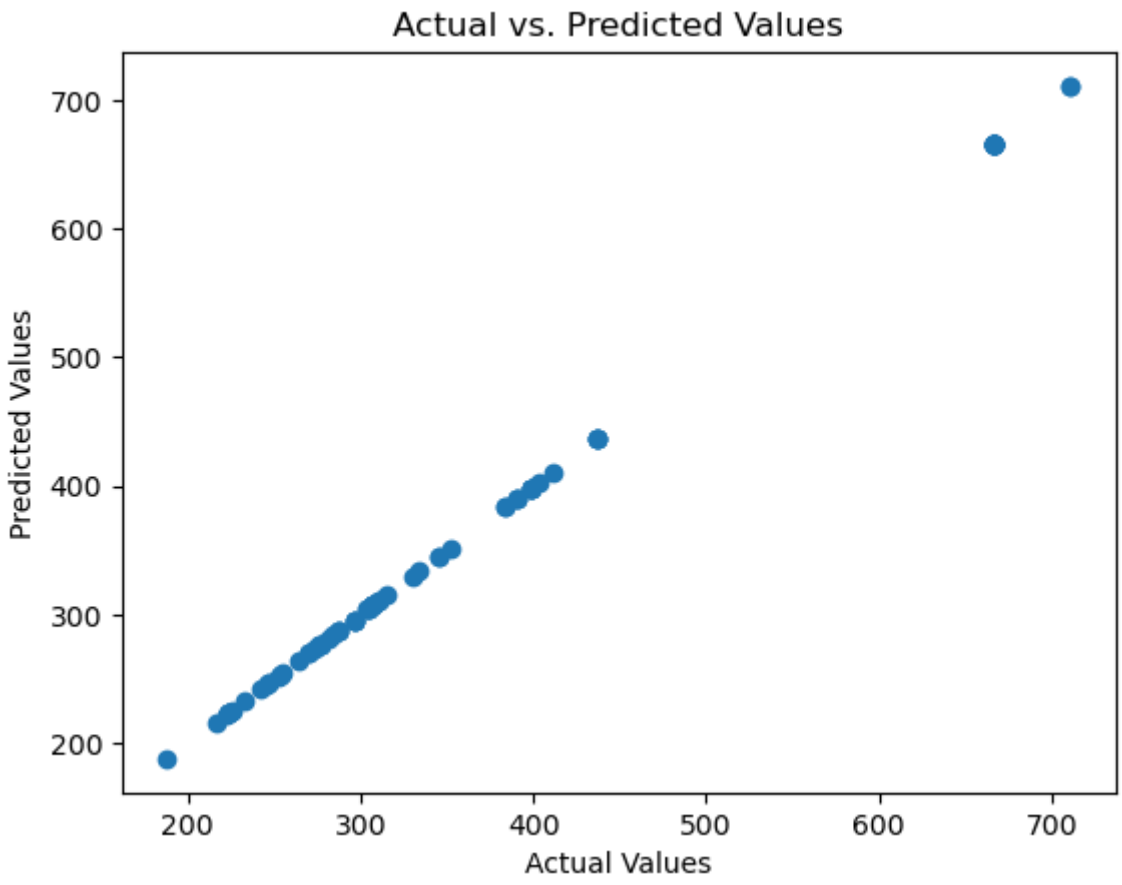


Underfitting

In [2]:

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error
import matplotlib.pyplot as plt

data = pd.read_csv(r"C:\Users\manju\Desktop\HousingData.csv")
X = data.drop('AGE', axis=1)
y = data['TAX']
X.fillna(X.mean(), inplace=True)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2
model = LinearRegression()
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
mse = mean_squared_error(y_test, y_pred)
print(f"Mean Squared Error: {mse}")
plt.scatter(y_test, y_pred)
plt.xlabel("Actual Values")
plt.ylabel("Predicted Values")
plt.title("Actual vs. Predicted Values")
plt.show()

```

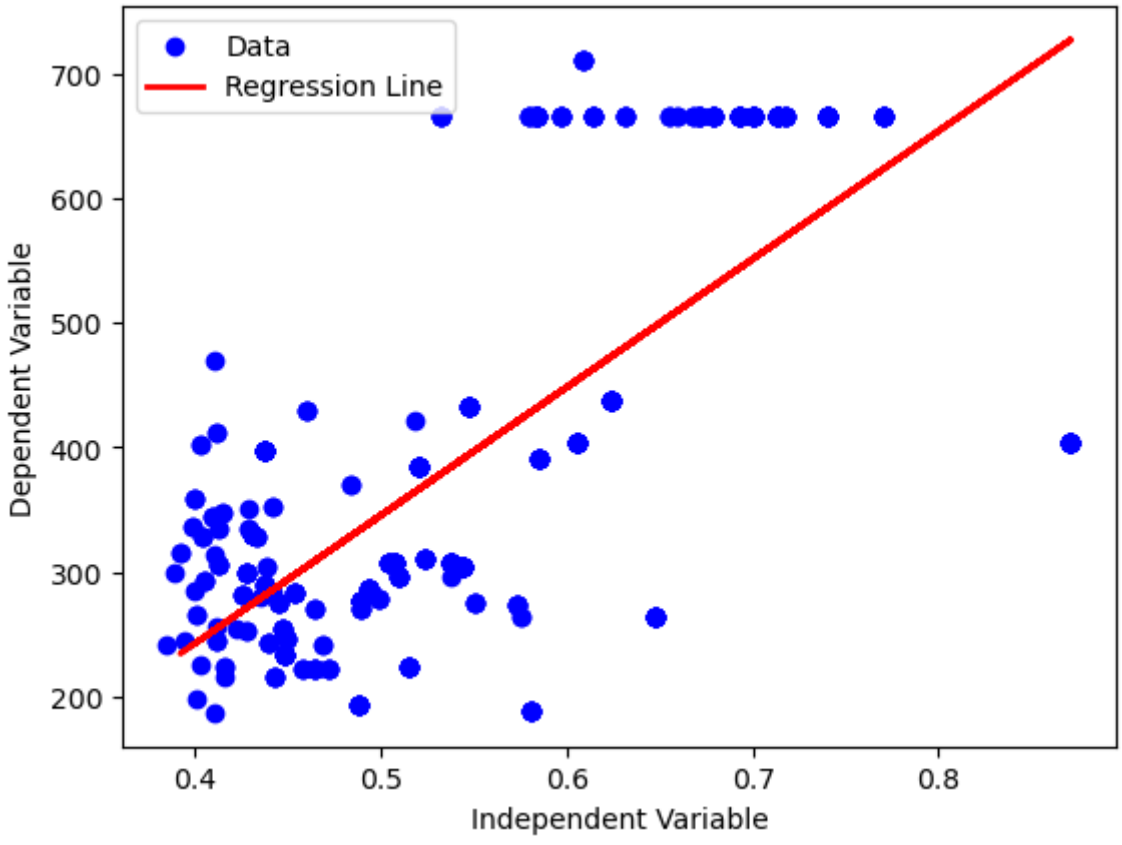
```
Mean Squared Error: 6.842486684721736e-27
```



simple linear regression

In [10]:

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
import matplotlib.pyplot as plt
data = pd.read_csv(r"C:\Users\manju\Desktop\HousingData.csv")
X = data[['NOX']]
y = data['TAX']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
model = LinearRegression()
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
plt.scatter(X, y, color='blue', label='Data')
plt.plot(X_test, y_pred, color='red', linewidth=2, label='Regression Lir
plt.xlabel('Independent Variable')
plt.ylabel('Dependent Variable')
plt.legend()
plt.show()
slope = model.coef_[0]
intercept = model.intercept_
print(f"Slope (m): {slope}")
print(f"Intercept (b): {intercept}")

```




```
Slope (m): 1027.9020979428083
Intercept (b): -168.55195983014949
```

Multiple linear regression

In [13]:

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score
data = pd.read_csv(r"C:\Users\manju\Desktop\HousingData.csv")
print(data.head())
print(data.info())
X = data[['NOX']]
y = data['TAX']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2
model = LinearRegression()
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)
print("Mean Squared Error:", mse)
print("R-squared:", r2)
print("Coefficients:", model.coef_)
print("Intercept:", model.intercept_)
```

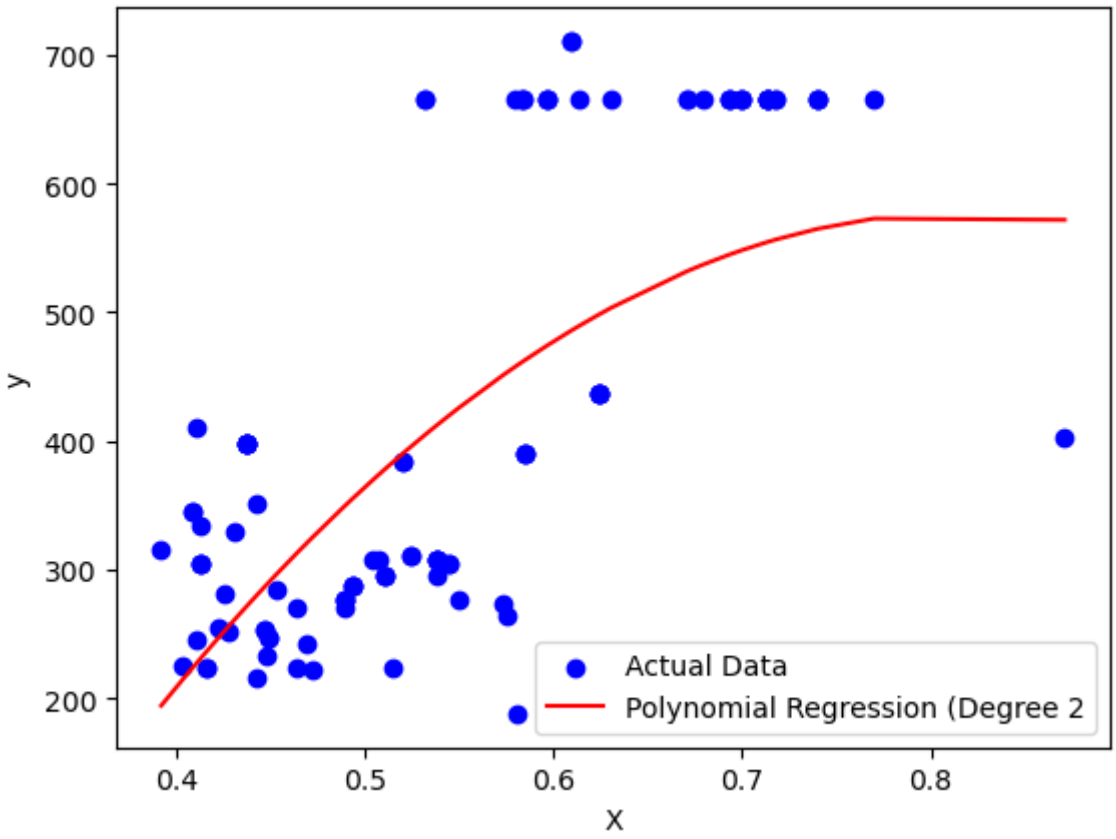
```
      CRIM    ZN  INDUS  CHAS    NOX     RM   AGE     DIS  RAD  TAX  PTRATI
O  \
0  0.00632  18.0   2.31   0.0  0.538  6.575  65.2  4.0900    1  296     15.
3
1  0.02731   0.0   7.07   0.0  0.469  6.421  78.9  4.9671    2  242     17.
8
2  0.02729   0.0   7.07   0.0  0.469  7.185  61.1  4.9671    2  242     17.
8
3  0.03237   0.0   2.18   0.0  0.458  6.998  45.8  6.0622    3  222     18.
7
4  0.06905   0.0   2.18   0.0  0.458  7.147  54.2  6.0622    3  222     18.
7

        B  LSTAT  MEDV
0  396.90   4.98  24.0
1  396.90   9.14  21.6
2  392.83   4.03  34.7
3  394.63   2.94  33.4
4  396.90    NaN  36.2
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 506 entries, 0 to 505
Data columns (total 14 columns):
 #   Column   Non-Null Count  Dtype
---  ------   --------------  -----
 0   CRIM     486 non-null    float64
 1   ZN       486 non-null    float64
 2   INDUS    486 non-null    float64
 3   CHAS     486 non-null    float64
 4   NOX      506 non-null    float64
 5   RM       506 non-null    float64
 6   AGE      486 non-null    float64
 7   DIS      506 non-null    float64
 8   RAD      506 non-null    int64
 9   TAX      506 non-null    int64
 10  PTRATIO  506 non-null    float64
 11  B        506 non-null    float64
 12  LSTAT    486 non-null    float64
 13  MEDV     506 non-null    float64
dtypes: float64(12), int64(2)
memory usage: 55.5 KB
None
Mean Squared Error: 15264.37775481554
R-squared: 0.510508842100377
Coefficients: [918.00440183]
Intercept: -106.82272875599114
```

polynomial linear regression

In [39]:

```
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 PolynomialFeatures
from sklearn.metrics import mean_squared_error, r2_score
import matplotlib.pyplot as plt
X = data[['NOX']]
y = data['TAX']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
degree = 2
poly_features = PolynomialFeatures(degree=degree)
X_train_poly = poly_features.fit_transform(X_train)
X_test_poly = poly_features.transform(X_test)
poly_reg = LinearRegression()
poly_reg.fit(X_train_poly, y_train)
y_pred = poly_reg.predict(X_test_poly)
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)
print("Mean Squared Error:", mse)
print("R-squared (R2) Score:", r2)
X_test_sorted, y_pred_sorted = zip(*sorted(zip(X_test.values, y_pred)))
plt.scatter(X_test, y_test, color='blue', label='Actual Data')
plt.plot(X_test_sorted, y_pred_sorted, color='red', label=f'Polynomial R
plt.xlabel('X')
plt.ylabel('y')
plt.legend()
plt.show()
```

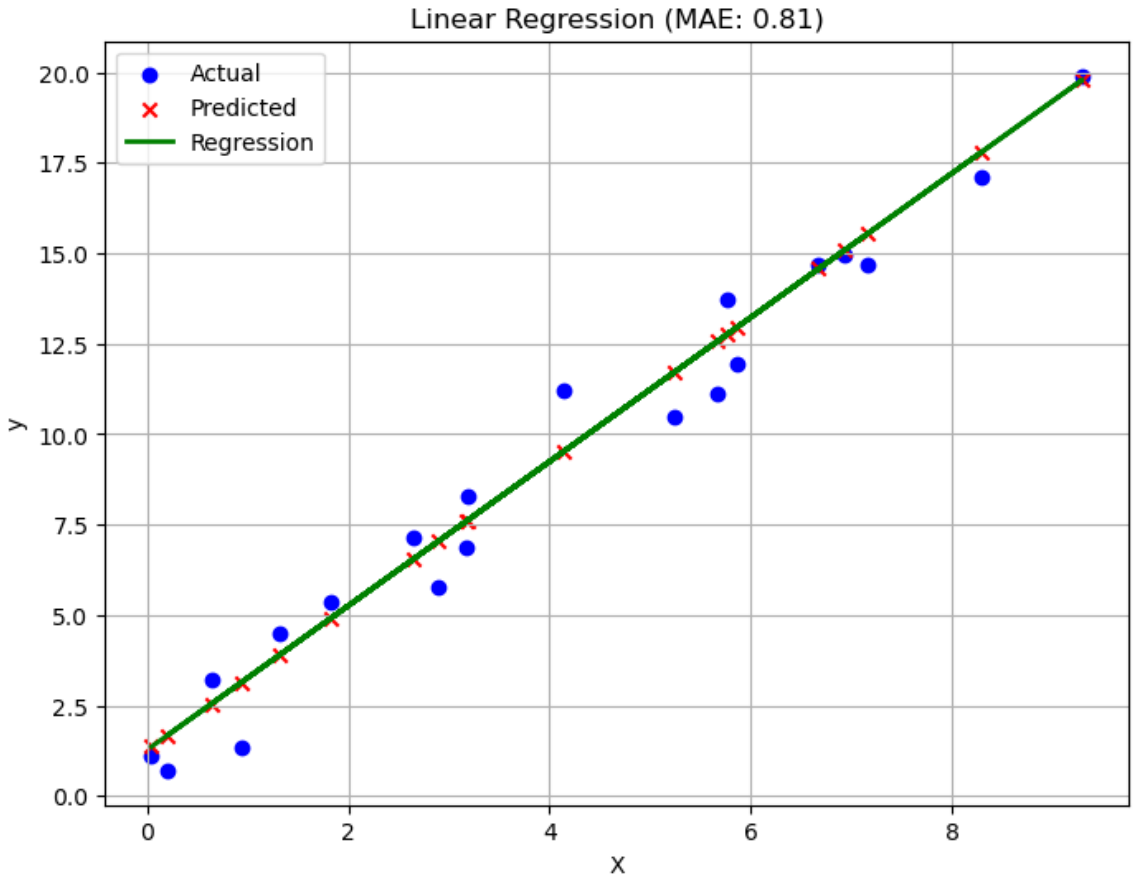
```
Mean Squared Error: 14546.715241982676
R-squared (R2) Score: 0.5335225187814907
```



MAE- mean absolute Error

In [7]:

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_absolute_error
np.random.seed(0)
X = np.random.rand(100, 1) * 10
y = 2 * X + 1 + np.random.randn(100, 1)
X_train, X_test, y_train, y_test = X[:80], X[80:], y[:80], y[80:]
model = LinearRegression()
model.fit(X_train, y_train)
predictions = model.predict(X_test)
mae = mean_absolute_error(y_test, predictions)
plt.figure(figsize=(8, 6))
plt.scatter(X_test, y_test, color='blue', label='Actual')
plt.scatter(X_test, predictions, color='red', marker='x', label='Predict
plt.plot(X_test, predictions, color='green', linewidth=2, label='Regress
plt.xlabel('X')
plt.ylabel('y')
plt.title(f'Linear Regression (MAE: {mae:.2f})')
plt.legend()
plt.grid()
plt.show()

```

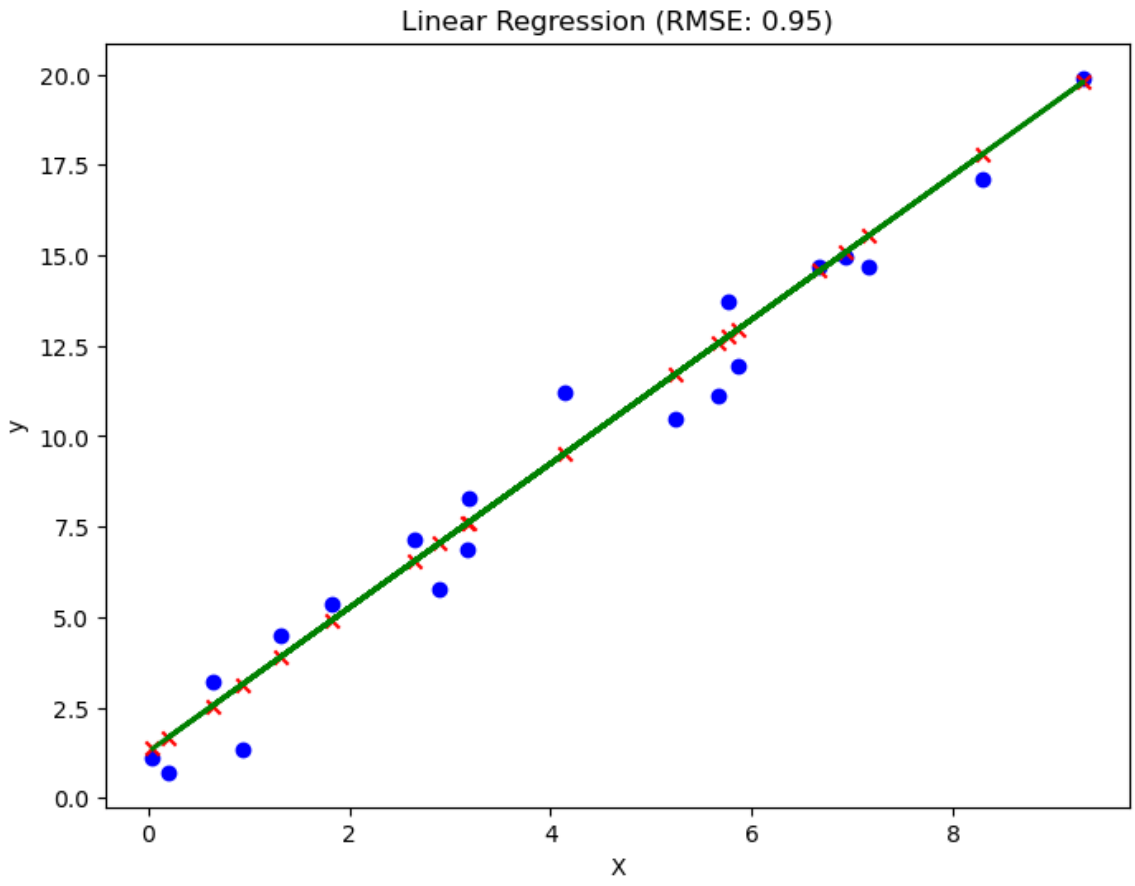


MSE- mean squard error

In [18]:

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error
np.random.seed(0)
X = np.random.rand(100, 1) * 10
y = 2 * X + 1 + np.random.randn(100, 1)
X_train, X_test, y_train, y_test = X[:80], X[80:], y[:80], y[80:]
model = LinearRegression()
model.fit(X_train, y_train)
predictions = model.predict(X_test)
squared_errors = (predictions - y_test) ** 2
rmse = np.sqrt(mean_squared_error(y_test, predictions))
plt.figure(figsize=(8, 6))
plt.scatter(X_test, y_test, color='blue', label='Actual')
plt.scatter(X_test, predictions, color='red', marker='x', label='Predict
plt.plot(X_test, predictions, color='green', linewidth=2, label='Regress
plt.xlabel('X')
plt.ylabel('y')
plt.title(f'Linear Regression (RMSE: {rmse:.2f})')
```

Out[18]: Text(0.5, 1.0, 'Linear Regression (RMSE: 0.95)')



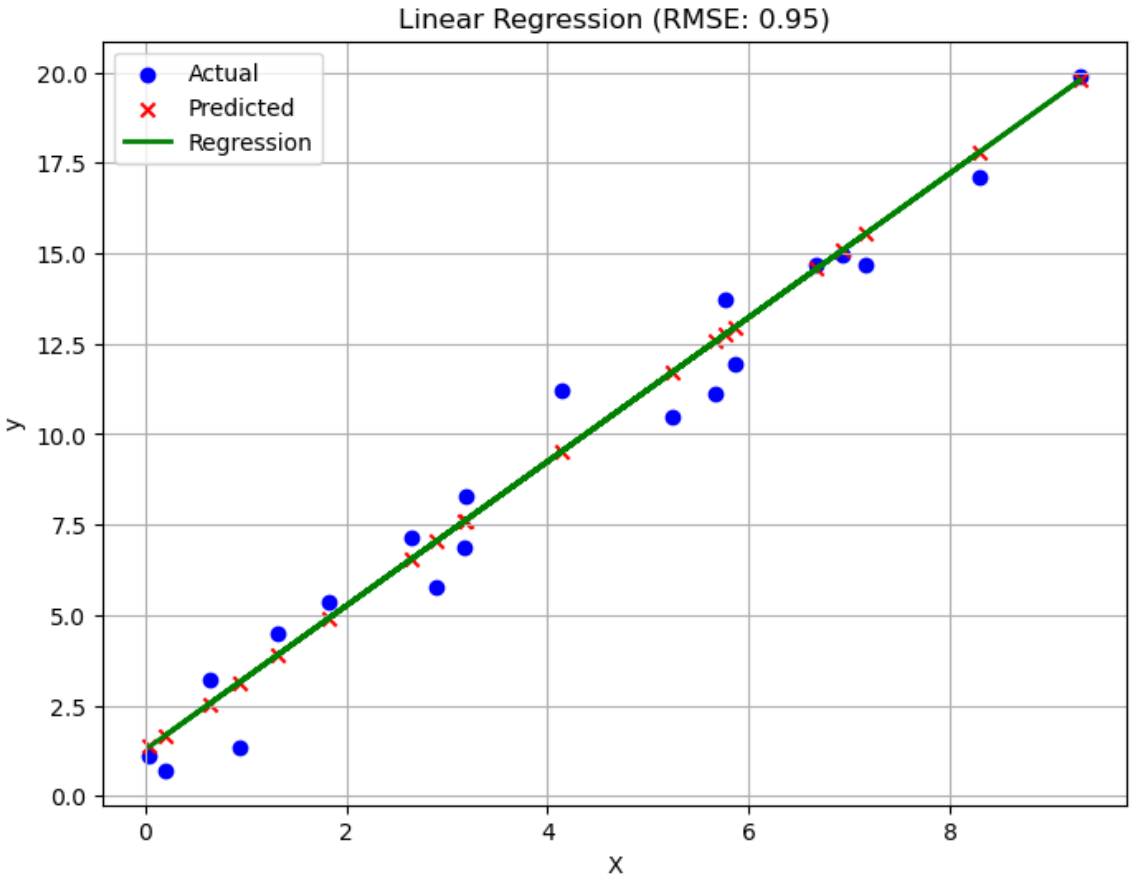
In [22]:

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squareed_error
np.random.seed(0)
X = np.random.rand(100, 1) * 10
y = 2 * X + 1 + np.random.randn(100, 1)
X_train, X_test, y_train, y_test = X[:80], X[80:], y[:80], y[80:]
model = LinearRegression()
model.fit(X_train, y_train)
predictions = model.predict(X_test)
mae = mean_absolute_error(y_test, predictions)
plt.figure(figsize=(8, 6))
plt.scatter(X_test, y_test, color='blue', label='Actual')
plt.scatter(X_test, predictions, color='red', marker='x', label='Predict
plt.plot(X_test, predictions, color='green', linewidth=2, label='Regress
plt.xlabel('X')
plt.ylabel('y')
plt.title(f'Linear Regression (MSE: {mse:.2f})')
plt.legend()
plt.grid()
plt.show()

```

```
---------------------------------------------------------------------------
ImportError                               Traceback (most recent call last)
Cell In[22], line 4
      2 import matplotlib.pyplot as plt
      3 from sklearn.linear_model import LinearRegression
----> 4 from sklearn.metrics import mean_squareed_error
      5 np.random.seed(0)
      6 X = np.random.rand(100, 1) * 10

ImportError: cannot import name 'mean_squareed_error' from 'sklearn.metric
s' (C:\Users\manju\anaconda3\Lib\site-packages\sklearn\metrics\__init__.py)
```

In [20]:

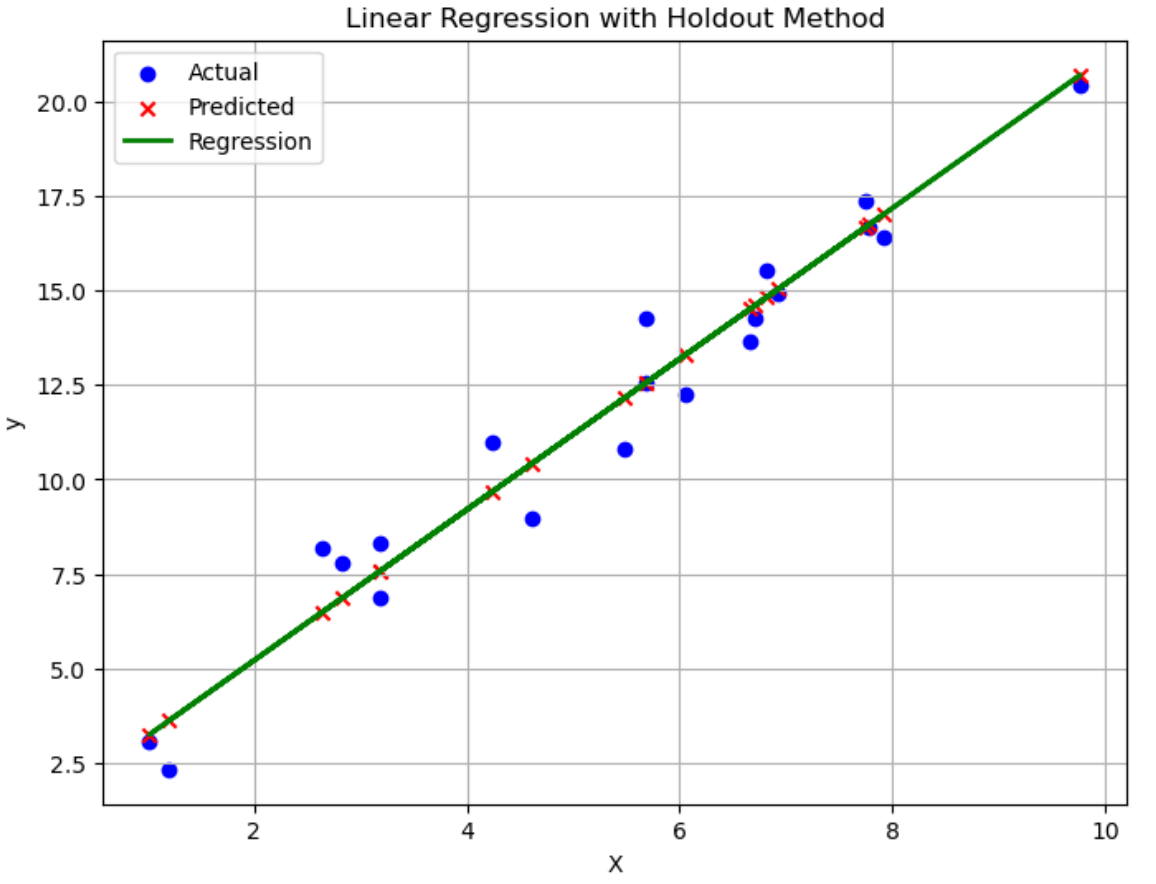
```python
#root mean squared error
import numpy as np
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error
np.random.seed(0)
X = np.random.rand(100, 1) * 10
y = 2 * X + 1 + np.random.randn(100, 1)
X_train, X_test, y_train, y_test = X[:80], X[80:], y[:80], y[80:]
model = LinearRegression()
model.fit(X_train, y_train)
predictions = model.predict(X_test)
squared_errors = (predictions - y_test) ** 2
rmse = np.sqrt(mean_squared_error(y_test, predictions))
plt.figure(figsize=(8, 6))
plt.scatter(X_test, y_test, color='blue', label='Actual')
plt.scatter(X_test, predictions, color='red', marker='x', label='Predict
plt.plot(X_test, predictions, color='green', linewidth=2, label='Regress
plt.xlabel('X')
plt.ylabel('y')
plt.title(f'Linear Regression (RMSE: {rmse:.2f})')
plt.legend()
plt.grid()
plt.show()


```



Hold out method

In [25]:

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
np.random.seed(0)
X = np.random.rand(100, 1) * 10
y = 2 * X + 1 + np.random.randn(100, 1)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
model = LinearRegression()
model.fit(X_train, y_train)
predictions = model.predict(X_test)
plt.figure(figsize=(8, 6))
plt.scatter(X_test, y_test, color='blue', label='Actual')
plt.scatter(X_test, predictions, color='red', marker='x', label='Predict
plt.plot(X_test, predictions, color='green', linewidth=2, label='Regress
plt.xlabel('X')
plt.ylabel('y')
plt.title('Linear Regression with Holdout Method')
plt.legend()
plt.grid()
plt.show()

```



In [29]:

```
# loading the breast_cancer dataset and finding the accuracy and classi
```

In [28]:

```
from sklearn.datasets import load_breast_cancer
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score, classification_report
breast_cancer = load_breast_cancer()
X = breast_cancer.data # Features
y = breast_cancer.target # Target variable
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
clf = DecisionTreeClassifier(random_state=42)
clf.fit(X_train, y_train)
y_pred = clf.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy: {accuracy:.2f}")
print("Classification Report:")
print(classification_report(y_test, y_pred))

```

```
Accuracy: 0.95
Classification Report:
              precision    recall  f1-score   support

           0       0.93      0.93      0.93        43
           1       0.96      0.96      0.96        71

    accuracy                           0.95       114
   macro avg       0.94      0.94      0.94       114
weighted avg       0.95      0.95      0.95       114
```

r-square

In [1]:

```python
#R-SQUARE
import numpy as np
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score
np.random.seed(0)
X = np.random.rand(100, 1) * 10
y = 2 * X + 1 + np.random.randn(100, 1)
X_train, X_test, y_train, y_test = X[:80], X[80:], y[:80], y[80:]
model = LinearRegression()
model.fit(X_train, y_train)
predictions = model.predict(X_test)
r_squared = r2_score(y_test, predictions)
print("X_train:", X_train)
print("y_train:", y_train)
print("X_test:", X_test)
print("y_test:", y_test)
print(f'R-squared: {r_squared:.2f}')

```

```
X_train: [[5.48813504]
 [7.15189366]
 [6.02763376]
 [5.44883183]
 [4.23654799]
 [6.45894113]
 [4.37587211]
 [8.91773001]
 [9.63662761]
 [3.83441519]
 [7.91725038]
 [5.2889492 ]
 [5.68044561]
 [9.25596638]
 [0.71036058]
 [0.871293  ]
 [0.20218397]
 [8.32619846]
 [7.78156751]
```

In [8]:

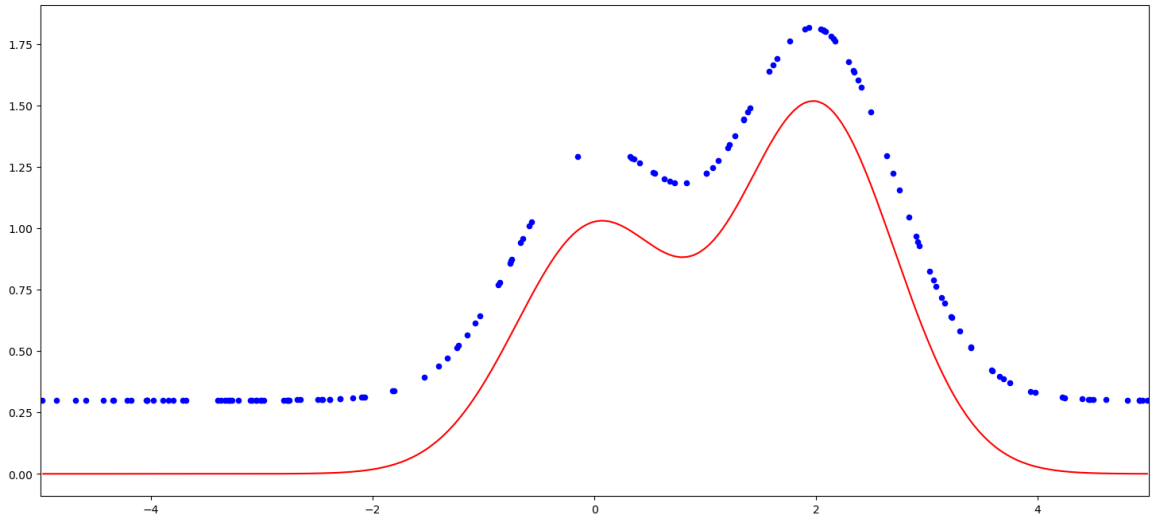
```
import numpy as np
import matplotlib.pyplot as plt
n_train = 150
n_test = 1000
noise = 0.1
def f(x):
 x = x.ravel()
 return np.exp(-x ** 2) + 1.5 * np.exp(-(x - 2) ** 2)
def generate(n_samples, noise):
 X = np.random.rand(n_samples) * 10 - 5
 X = np.sort(X).ravel()
 y = np.exp(-X ** 2) + 1.5 * np.exp(-(X - 2) ** 2) + np.random.normal(0
 X = X.reshape((n_samples, 1))
 return X, y
X_train, y_train = generate(n_samples=n_train, noise=noise)
X_test, y_test = generate(n_samples=n_test, noise=noise)
```

In [ ]:

```
1
```

In [7]:

```
plt.figure(figsize=(18, 8))
plt.plot(X_test, f(X_test), "r")
plt.scatter(X_train, y_train, c="b", s=20)
plt.xlim([-5, 5])
plt.show()
```

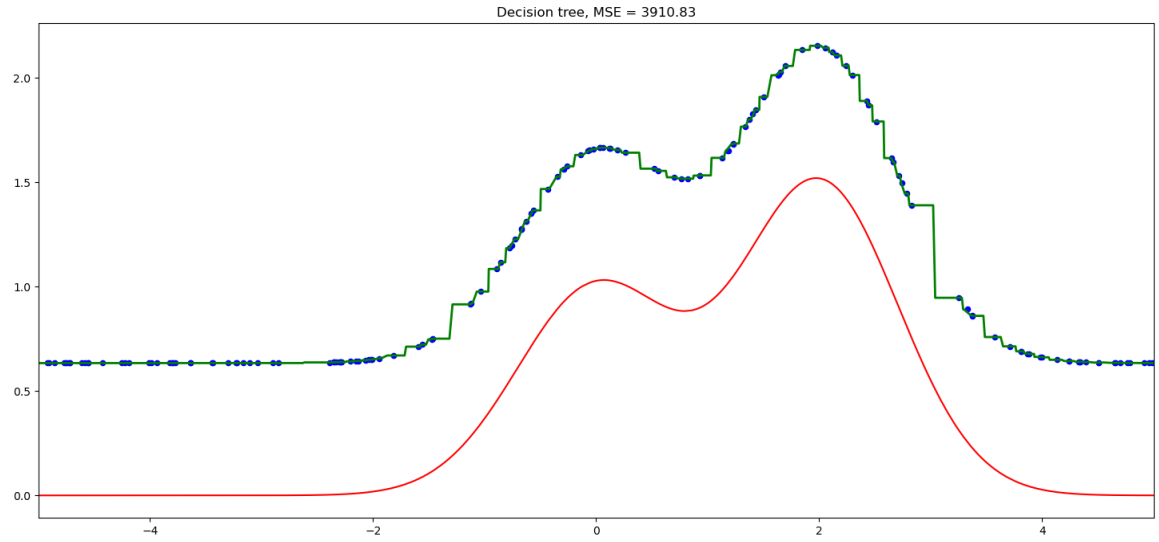


decision tree Regresson

In [9]:

```
#DecisionTreeRegressor
from sklearn.tree import DecisionTreeRegressor
dtree = DecisionTreeRegressor().fit(X_train, y_train)
d_predict = dtree.predict(X_test)
plt.figure(figsize=(18, 8))
plt.plot(X_test, f(X_test), "r")
plt.scatter(X_train, y_train, c="b", s=20)
plt.plot(X_test, d_predict, "g", lw=2)
plt.xlim([-5, 5])
plt.title("Decision tree, MSE = %.2f"
% np.sum((y_test - d_predict) ** 2))
```

Out[9]: Text(0.5, 1.0, 'Decision tree, MSE = 3910.83')

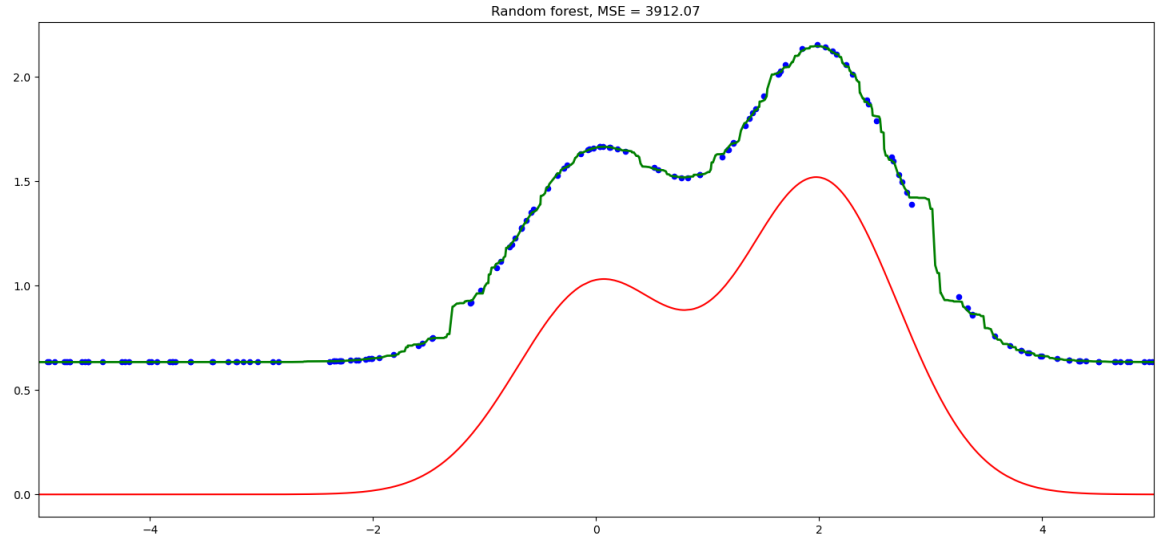


random forest algorithm

In [11]:

```
from sklearn.ensemble import RandomForestRegressor
rfr = RandomForestRegressor(n_estimators=1000).fit(X_train, y_train)
rf_predict = rfr.predict(X_test)
plt.figure(figsize=(18, 8))
plt.plot(X_test, f(X_test), "r")
plt.scatter(X_train, y_train, c="b", s=20)
plt.plot(X_test, rf_predict, "g", lw=2)
plt.xlim([-5, 5])
plt.title("Random forest, MSE = %.2f" % np.sum((y_test - rf_predict) **

```

Out[11]: Text(0.5, 1.0, 'Random forest, MSE = 3912.07')



In [ ]:

```
1
```