```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

```
df = pd.read_csv('insurance.csv')
df.head()
```

	age	sex	bmi	children	smoker	region	charges
0	19	female	27.900	0	yes	southwest	16884.92400
1	18	male	33.770	1	no	southeast	1725.55230
2	28	male	33.000	3	no	southeast	4449.46200
3	33	male	22.705	0	no	northwest	21984.47061
4	32	male	28.880	0	no	northwest	3866.85520

```
df = pd.read_csv('insurance.csv')
df.head()
```

	age	sex	bmi	children	smoker	region	charges
0	19	female	27.900	0	yes	southwest	16884.92400
1	18	male	33.770	1	no	southeast	1725.55230
2	28	male	33.000	3	no	southeast	4449.46200
3	33	male	22.705	0	no	northwest	21984.47061
4	32	male	28.880	0	no	northwest	3866.85520

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1338 entries, 0 to 1337
Data columns (total 7 columns):
 #   Column    Non-Null Count  Dtype  
---  ------    --------------  -----  
 0   age       1338 non-null   int64  
 1   sex       1338 non-null   object 
 2   bmi       1338 non-null   float64
 3   children  1338 non-null   int64  
 4   smoker    1338 non-null   object 
 5   region    1338 non-null   object 
 6   charges   1338 non-null   float64
dtypes: float64(2), int64(2), object(3)
memory usage: 73.3+ KB
```

```
df.describe()
```

	age	bmi	children	charges
count	1338.000000	1338.000000	1338.000000	1338.000000
mean	39.207025	30.663397	1.094918	13270.422265
std	14.049960	6.098187	1.205493	12110.011237

```
print(df.sex.value_counts(),'\n',df.smoker.value_counts(),'\n',df.region.value_counts())
```

```
male      676
female    662
Name: sex, dtype: int64
 no     1064
yes     274
Name: smoker, dtype: int64
 southeast    364
southwest    325
northwest    325
northeast    324
Name: region, dtype: int64
```

```
#changing categorical variables to numerical
df['sex'] = df['sex'].map({'male':1,'female':0})
df['smoker'] = df['smoker'].map({'yes':1,'no':0})
df['region'] = df['region'].map({'southwest':0,'southeast':1,'northwest':2,'northeast':3})
```

```
df.head(10)
```

	age	sex	bmi	children	smoker	region	charges
0	19	0	27.900	0	1	0	16884.92400
1	18	1	33.770	1	0	1	1725.55230
2	28	1	33.000	3	0	1	4449.46200
3	33	1	22.705	0	0	2	21984.47061
4	32	1	28.880	0	0	2	3866.85520
5	31	0	25.740	0	0	1	3756.62160
6	46	0	33.440	1	0	1	8240.58960
7	37	0	27.740	3	0	2	7281.50560
8	37	1	29.830	2	0	3	6406.41070
9	60	0	25.840	0	0	2	28923.13692

bold text

Train test split

```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(df.drop('charges',axis=1), df['charges'], test_
```

Linear Regression

```
from sklearn.linear_model import LinearRegression
lr = LinearRegression()
lr
```

```
▾ LinearRegression
LinearRegression()
```

```
#model training
lr.fit(x_train,y_train)
#model accuracy
lr.score(x_train,y_train)
```

```
0.7368306228430945
```

```
#model prediction
y_pred = lr.predict(x_test)
```

Polynomial Linear Regression

```
from sklearn.preprocessing import PolynomialFeatures
poly_reg = PolynomialFeatures(degree=2)
poly_reg
```

```
▾ PolynomialFeatures
PolynomialFeatures()
```

```
#transforming the features to higher degree
x_train_poly = poly_reg.fit_transform(x_train)
#splitting the data
x_train, x_test, y_train, y_test = train_test_split(x_train_poly, y_train, test_size=0.2, random_s
```

```
plr = LinearRegression()
#model training
plr.fit(x_train,y_train)
#model accuracy
plr.score(x_train,y_train)
```

```
0.836373486593943
```

```
#model prediction
y_pred = plr.predict(x_test)
```

Decision Tree Regression

```
from sklearn.tree import DecisionTreeRegressor
dtree = DecisionTreeRegressor()
dtree
```

```
▾ DecisionTreeRegressor
DecisionTreeRegressor()
```

```
#model training
dtree.fit(x_train,y_train)
#model accuracy
dtree.score(x_train,y_train)
```

```
0.9993688476658964
```

```
#model prediction
```

```
#model prediction
dtree_pred = dtree.predict(x_test)
```

ID3

```
!pip install decision-tree-id3
```

```
Requirement already satisfied: decision-tree-id3 in /usr/local/lib/python3.10/dist-packages (0.
Requirement already satisfied: nose>=1.1.2 in /usr/local/lib/python3.10/dist-packages (from dec
Requirement already satisfied: scikit-learn>=0.17 in /usr/local/lib/python3.10/dist-packages (f
Requirement already satisfied: numpy>=1.6.1 in /usr/local/lib/python3.10/dist-packages (from de
Requirement already satisfied: scipy>=1.3.2 in /usr/local/lib/python3.10/dist-packages (from sc
Requirement already satisfied: joblib>=1.1.1 in /usr/local/lib/python3.10/dist-packages (from s
Requirement already satisfied: threadpoolctl>=2.0.0 in /usr/local/lib/python3.10/dist-packages
```

```
import six
import sys
sys.modules['sklearn.externals.six'] = six
```

```
from id3 import Id3Estimator

# Create an instance of the ID3 estimator
id3_tree = Id3Estimator()

# Fit the ID3 tree to your data
id3_tree.fit(x_train, y_train)

# Make predictions
id3_predictions = id3_tree.predict(x_test)
```

Random Forest Regression

```
#random forest regressor
from sklearn.ensemble import RandomForestRegressor
rf = RandomForestRegressor(n_estimators=100)
rf
```

```
▾ RandomForestRegressor
RandomForestRegressor()
```

```
RandomForestRegressor()
#model training
rf.fit(x_train,y_train)
#model accuracy
rf.score(x_train,y_train)
```

```
0.973598568706551
```

```
#model prediction
rf_pred = rf.predict(x_test)
```

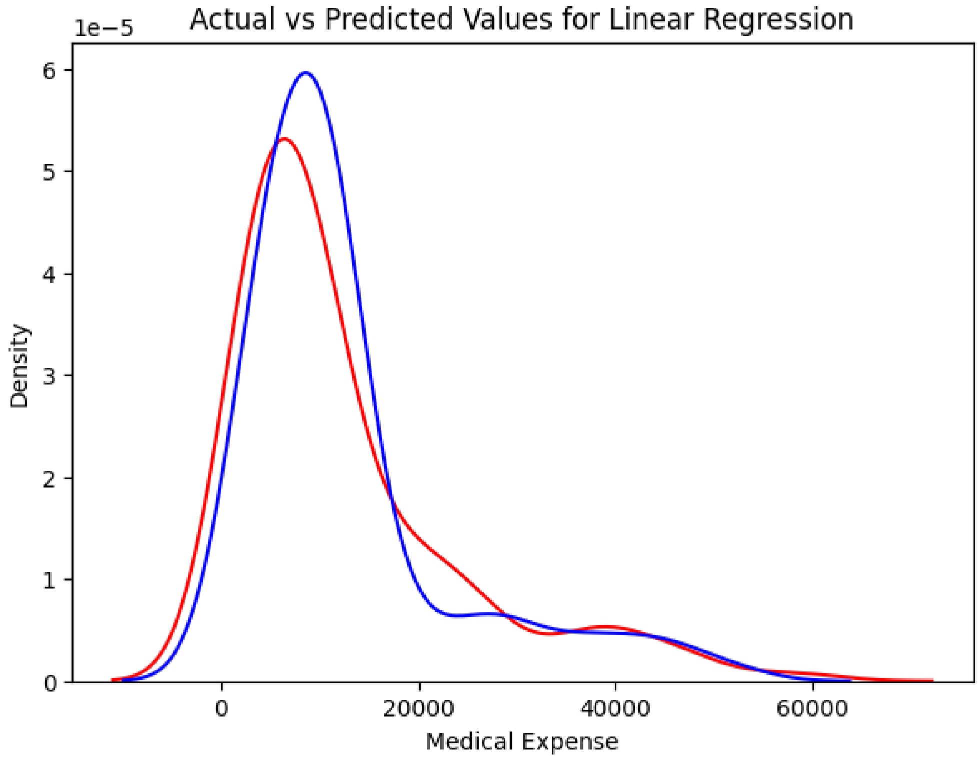
Model Evaluation

```
from sklearn.metrics import mean_squared_error,mean_absolute_error,r2_score
```

LINEAR REGRESSION

```
#distribution of actual and predicted values
plt.figure(figsize=(7,5))
ax1 = sns.distplot(y_test,hist=False,color='r',label='Actual Value')
sns.distplot(y_pred,hist=False,color='b',label='Predicted Value',ax=ax1)
plt.title('Actual vs Predicted Values for Linear Regression')
plt.xlabel('Medical Expense')
plt.show()
```

```
<ipython-input-31-2d0e63236188>:3: 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 `kdeplot` (an axes-level function for kernel density

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  ax1 = sns.distplot(y_test,hist=False,color='r',label='Actual Value')
<ipython-input-31-2d0e63236188>:4: 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 `kdeplot` (an axes-level function for kernel density

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(y_pred,hist=False,color='b',label='Predicted Value',ax=ax1)
```



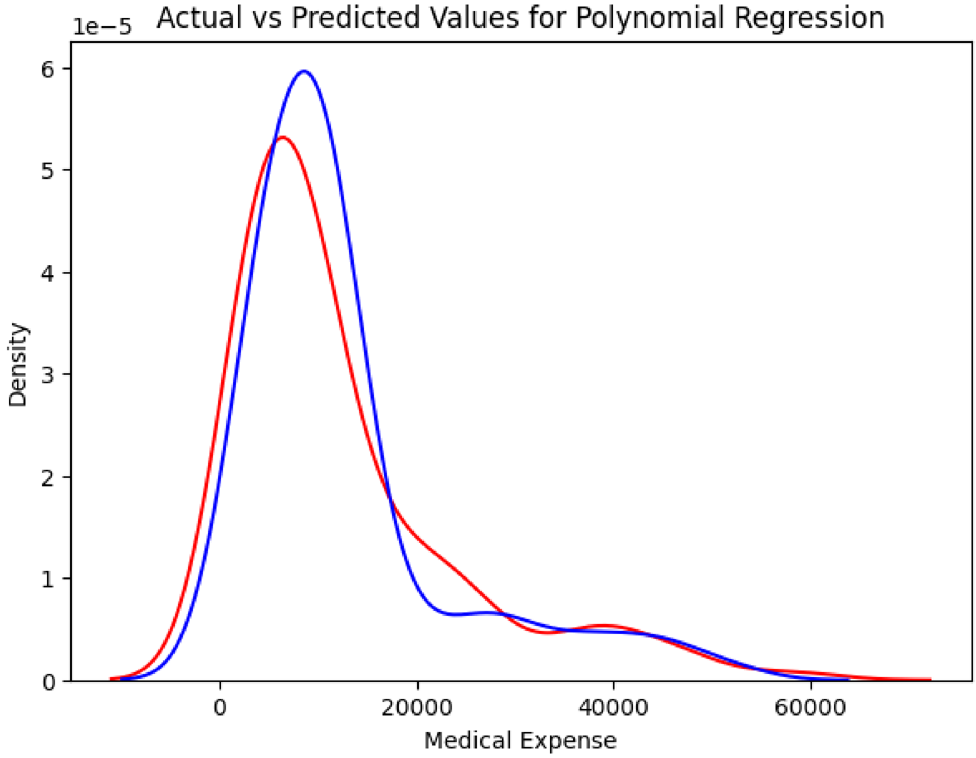

```
print('MAE:', mean_absolute_error(y_test, y_pred))
print('MSE:', mean_squared_error(y_test, y_pred))
print('RMSE:', np.sqrt(mean_squared_error(y_test, y_pred)))
print('R2 Score:', r2_score(y_test, y_pred))
```

```
MAE: 3016.8193118925233
MSE: 24705741.734187007
RMSE: 4970.48707212754
R2 Score: 0.8207480676082507
```

POLYNOMIAL REGRESSION

```
#acutal vs predicted values for polynomial regression
plt.figure(figsize=(7,5))
ax1 = sns.distplot(y_test,hist=False,color='r',label='Actual Value')
sns.distplot(y_pred,hist=False,color='b',label='Predicted Value',ax=ax1)
plt.title('Actual vs Predicted Values for Polynomial Regression')
plt.xlabel('Medical Expense')
plt.show()
```

```
<ipython-input-33-7a574536b1bb>:3: 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 `kdeplot` (an axes-level function for kernel density

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  ax1 = sns.distplot(y_test,hist=False,color='r',label='Actual Value')
<ipython-input-33-7a574536b1bb>:4: 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 `kdeplot` (an axes-level function for kernel density

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(y_pred,hist=False,color='b',label='Predicted Value',ax=ax1)
```




```
print('MAE:', mean_absolute_error(y_test, y_pred))
print('MSE:', mean_squared_error(y_test, y_pred))
print('RMSE:', np.sqrt(mean_squared_error(y_test, y_pred)))
print('R2 Score:', r2_score(y_test, y_pred))
```

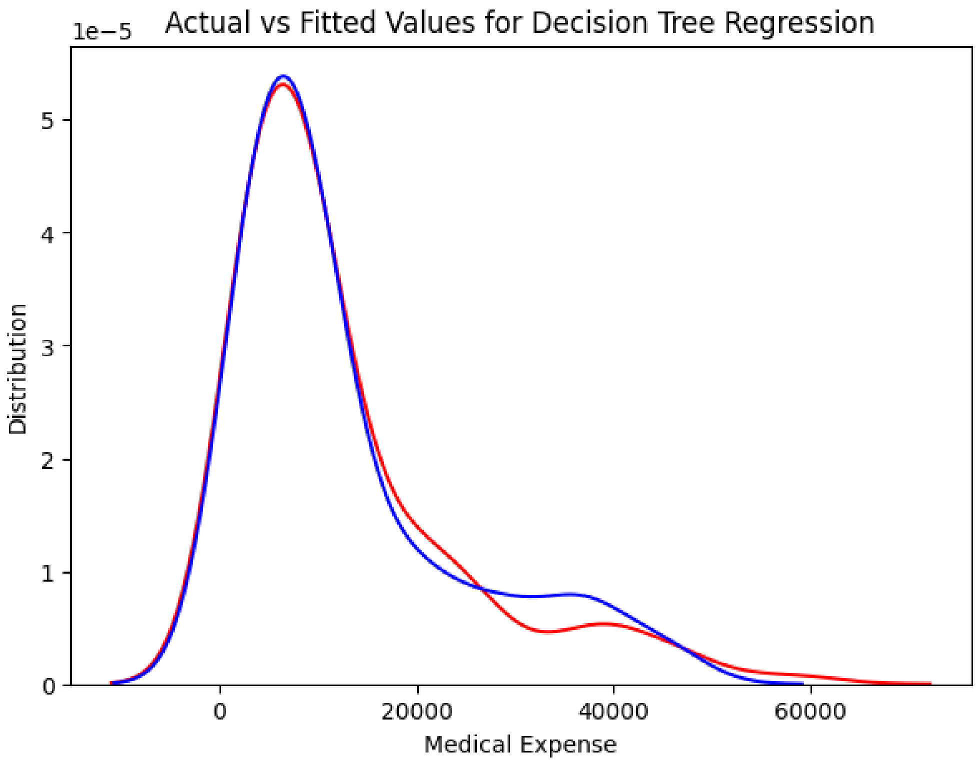
```
MAE: 3016.8193118925233
MSE: 24705741.734187007
```

```
RMSE: 4970.48707212754
R2 Score: 0.8207480676082507
```

DECISSION TREE

```
#distribution plot of actual and predicted values
plt.figure(figsize=(7,5))
ax = sns.distplot(y_test, hist=False, color="r", label="Actual Value")
sns.distplot(dtree_pred, hist=False, color="b", label="Fitted Values" , ax=ax)
plt.title('Actual vs Fitted Values for Decision Tree Regression')
plt.xlabel('Medical Expense')
plt.ylabel('Distribution')
plt.show()
```

```
<ipython-input-35-46f60f40ec0e>:3: 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 `kdeplot` (an axes-level function for kernel density

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  ax = sns.distplot(y_test, hist=False, color="r", label="Actual Value")
<ipython-input-35-46f60f40ec0e>:4: 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 `kdeplot` (an axes-level function for kernel density

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(dtree_pred, hist=False, color="b", label="Fitted Values" , ax=
```



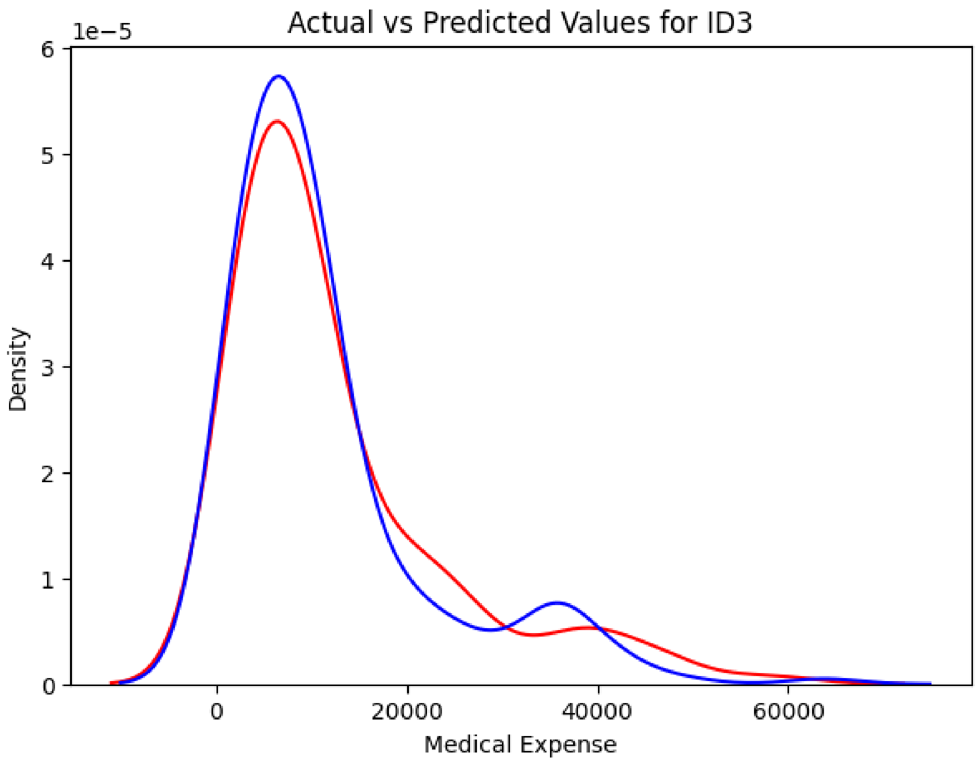
```
print('MAE:', mean_absolute_error(y_test, dtree_pred))
print('MSE:', mean_squared_error(y_test, dtree_pred))
print('RMSE:', np.sqrt(mean_squared_error(y_test, dtree_pred)))
print('Accuracy:', dtree.score(x_test,y_test))
```

```
MAE: 3333.450397056075
MSE: 50502211.929683186
RMSE: 7106.490830901225
Accuracy: 0.6335823803287539
```

ID3

```
plt.figure(figsize=(7,5))
ax1 = sns.distplot(y_test,hist=False,color='r',label='Actual Value')
sns.distplot(id3_predictions,hist=False,color='b',label='Predicted Value',ax=ax1)
plt.title('Actual vs Predicted Values for ID3')
plt.xlabel('Medical Expense')
plt.show()
```

```
<ipython-input-37-25c3e5658fb1>: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 `kdeplot` (an axes-level function for kernel density

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  ax1 = sns.distplot(y_test,hist=False,color='r',label='Actual Value')
<ipython-input-37-25c3e5658fb1>:3: 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 `kdeplot` (an axes-level function for kernel density

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(id3_predictions,hist=False,color='b',label='Predicted Value',a
```



```
from sklearn.metrics import mean_squared_error, r2_score

# Assuming y_test and id3_predictions are your true and predicted values in a regression problem
mse = mean_squared_error(y_test, id3_predictions)
r2 = r2_score(y_test, id3_predictions)

print("Mean Squared Error:", mse)
print("R-squared:", r2)
```

```
Mean Squared Error: 152902599.817644
R-squared: -0.10938124343413658
```

RANDOM FOREST

```
#distribution plot of actual and predicted values
plt.figure(figsize=(7,5))
ax = sns.distplot(y_test, hist=False, color="r", label="Actual Value")
sns.distplot(rf_pred, hist=False, color="b", label="Fitted Values" , ax=ax)
plt.title('Actual vs Fitted Values for Random Forest Regressor')
plt.xlabel('Medical Expense')
plt.ylabel('Distribution')
plt.show()
```

```
<ipython-input-38-255136d82566>:3: 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
```

```
print('MAE:', mean_absolute_error(y_test, rf_pred))
print('MSE:', mean_squared_error(y_test, rf_pred))
print('RMSE:', np.sqrt(mean_squared_error(y_test, rf_pred)))
print('Accuracy:', rf.score(x_test,y_test))
```

```
MAE: 2835.858837917913
MSE: 26936858.866949964
RMSE: 5190.073108054448
Accuracy: 0.8045602493373794
```

```
similar flexibility) or `kdeplot` (an axes-level function for kernel density

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(rf_pred, hist=False, color="b", label="Fitted Values" , ax=ax)
```