1. **Identify your problem statement**
    - **Problem Statement**: To Predict the insurance charges based on the parameters “age”, “sex”, “bmi”, “children” and “smoker”.
2. Tell basic info about the dataset (Total number of rows, columns)
    - The input table has 1338 rows and 6 columns
        i. Age – Integer
        ii. Sex – Male or Female
        iii. BMI – Float
        iv. Children - Integer (whole)
        v. Smoker – Yes or No
        vi. Charges – Float, needs cleaning as some of the values are not in proper format

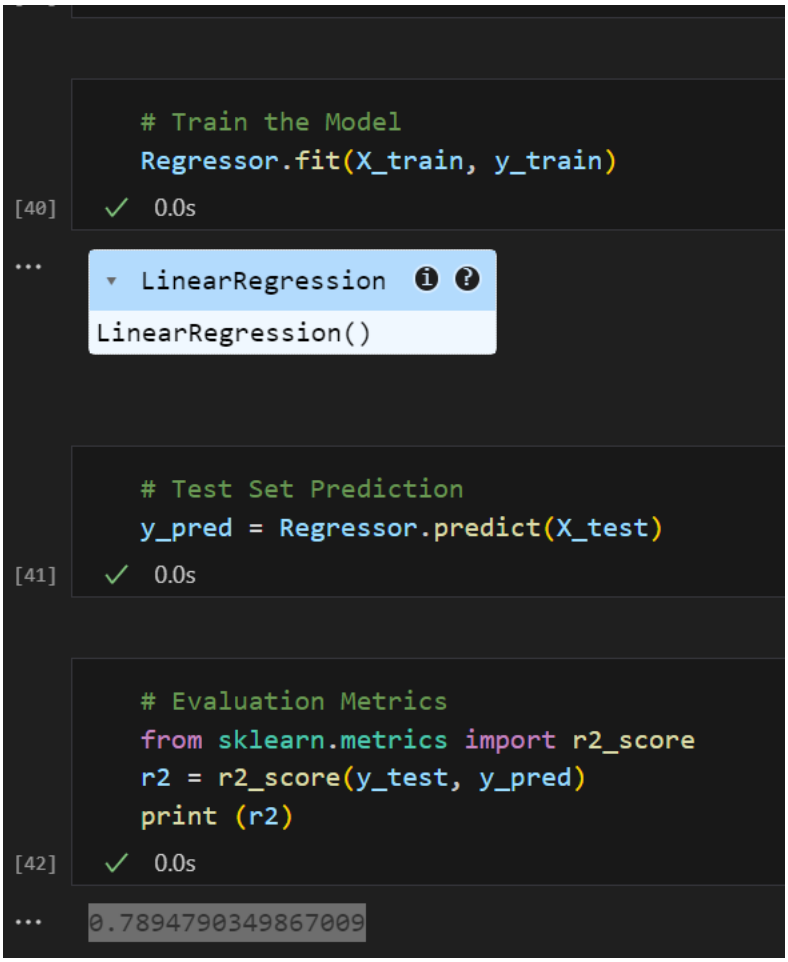
```
dataset
✓ 0.0s
```

	age	sex	bmi	children	smoker	charges
0	19	female	27.900	0	yes	16884.92400
1	18	male	33.770	1	no	1725.55230
2	28	male	33.000	3	no	4449.46200
3	33	male	22.705	0	no	21984.47061
4	32	male	28.880	0	no	3866.85520
...	...	...	...	...	...	...
1333	50	male	30.970	3	no	10600.54830
1334	18	female	31.920	0	no	2205.98080
1335	18	female	36.850	0	no	1629.83350
1336	21	female	25.800	0	no	2007.94500
1337	61	female	29.070	0	yes	29141.36030

1338 rows × 6 columns

3. Mention the pre-processing method if you’re doing any (like converting string to number – nominal data)
    - Data type of “charges” should be standardized
4. Develop a good model with r2_score. You can use any machine learning algorithm; you can create many models. Finally, you have to come up with final model.

Multiple Linear Regression Results



SVM

```
from sklearn.svm import SVR
Regressor = SVR(kernel='rbf', C=1000, gamma=0.01, epsilon=.5)
```

✓ 0.0s

```
# Train the Model
Regressor.fit(X_train, y_train)
```

✓ 0.0s

SVR

```
SVR(C=1000, epsilon=0.5, gamma=0.01)
```

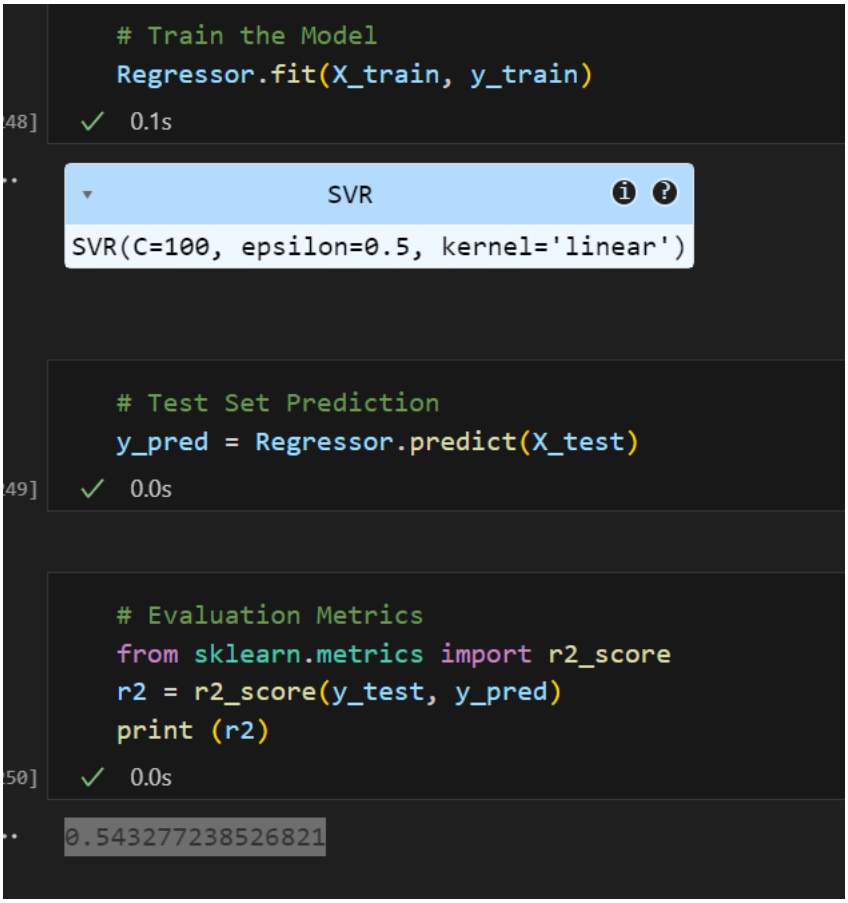
```
# Test Set Prediction
y_pred = Regressor.predict(X_test)
```

✓ 0.0s

```
# Evaluation Metrics
from sklearn.metrics import r2_score
r2 = r2_score(y_test, y_pred)
print (r2)
```

✓ 0.0s

```
-0.08064775139253766
```



Decision Tree

```
# Train the Model
Regressor.fit(X_train, y_train)
```

[404] ✓ 0.0s

```
DecisionTreeRegressor
DecisionTreeRegressor(max_depth=3, min_samples_split=5, random_state=0)
```

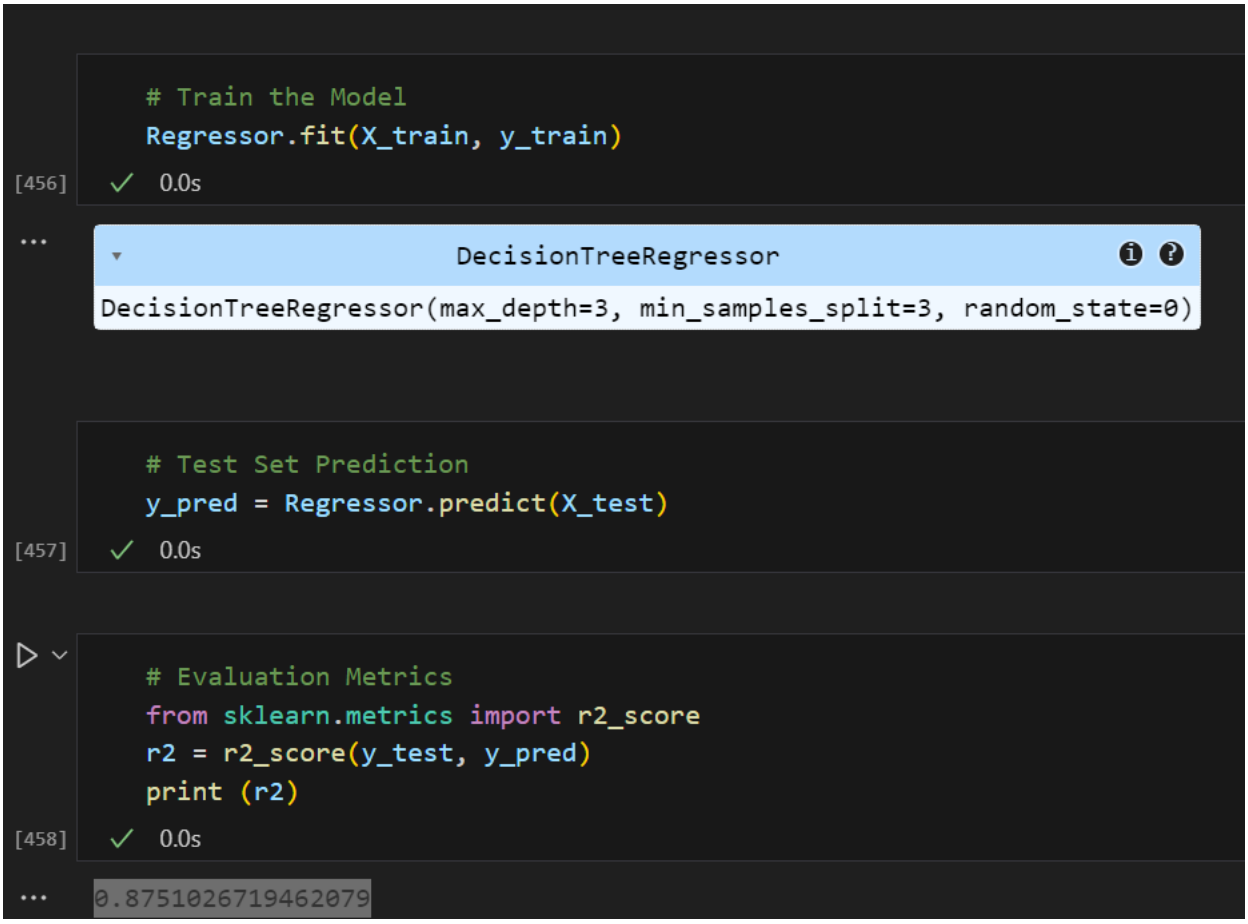
```
# Test Set Prediction
y_pred = Regressor.predict(X_test)
```

[405] ✓ 0.0s

```
# Evaluation Metrics
from sklearn.metrics import r2_score
r2 = r2_score(y_test, y_pred)
print (r2)
```

[406] ✓ 0.0s

```
0.8751026719462079
```



Random Forest

```
# Train the Model
Regressor.fit(X_train, y_train)
```
✓ 1.9s

RandomForestRegressor

```
RandomForestRegressor(n_estimators=1000, random_state=0)
```

```
# Test Set Prediction
y_pred = Regressor.predict(X_test)
```
✓ 0.0s

```
# Evaluation Metrics
from sklearn.metrics import r2_score
r2 = r2_score(y_test, y_pred)
print (r2)
```
✓ 0.0s

```
0.8541778123151671
```

RandomForestRegressor

```
RandomForestRegressor(max_depth=10, max_features='sqrt', n_estimators=1000,
                      random_state=10)
```

```
# Test Set Prediction
y_pred = Regressor.predict(X_test)
```
[516] ✓ 0.0s

```
# Evaluation Metrics
from sklearn.metrics import r2_score
r2 = r2_score(y_test, y_pred)
print (r2)
```
[517] ✓ 0.0s

```
0.8794110308126855
```

5. All the research values (r2_score of the models) should be documented. (You can make tabulation or screenshot of the results.)

Multiple Linear Regression	SVM			Decision Tree		Random Forest	
r2_score	kernel	Parameters	r2_score	Parameters	R^2 Score	Parameters	R^2 Score
0.789479035	rbf	SVR(C=1000, epsilon=1, gamma=10	-0.0869227	random_state=0, max_depth=5, min_samples_split=5	0.8247593	n_estimators=100, random_state=0	0.8538
	rbf	SVR(C=10, epsilon=0.5, gamma=1)	-0.0893862	max_depth=3, min_samples_split=5, random_state=0	0.87510267	n_estimators=1000, random_state=0	0.8542
	linear	C=1.0, epsilon=0.1	-0.1116613	max_depth=3, min_samples_split=5, random_state=1	0.87510267	n_estimators=1000, random_state=10, max_depth=10, min_samples_split=2, min_samples_leaf=1, max_features='sqrt'	0.8794
	linear	C=100, epsilon=0.5	0.54327724	max_depth=3, min_samples_split=3, random_state=0	0.87510267	n_estimators=1000, random_state=10, max_depth=10, min_samples_split=5, min_samples_leaf=1, max_features='sqrt'	0.8828
	linear	C=1000, epsilon=0.5	0.63403982			n_estimators=1000, random_state=10, max_depth=10, min_samples_split=5, min_samples_leaf=1, max_features='log'	0.8828
	linear	C=1000, epsilon=1	0.63404343			n_estimators=10000, random_state=10, max_depth=10, min_samples_split=5, min_samples_leaf=1, max_features='log'	0.8831
	linear	C=10000, epsilon=1	0.74448529				
	poly	degree=2, C=1000, epsilon=0.1	-0.1071166				
	poly	degree=5, C=1000, epsilon=0.5	0.22356695				
	sigmoid	C=1000, gamma=0.1, epsilon=.1	-0.089709				

6. Mention your final model, justify why u have chosen the same

   Chosen the Random Forest algorithm with the parameters *max_depth=10, max_features='sqrt', min_samples_split=5, n_estimators=10000, random_state=10* yielded the highest R_Squared value.

7. Name the .pynb file properly and upload in GitHub.
   - Regression assignment.pynb https://github.com/saravananjay/Machine-Learning/blob/main/Regression%20Assignment.ipynb