Machine Learning

Medical Cost Personal

Project

Hussein Hesham Hussein

20190183

Nour El-Din Ahmed Ezzat

20190593

3- Medical Cost Personal

It is a dataset for regression tasks. It consists of 1300+ records containing persons medical data and the target is "charge" column. The goal is make a model that fits these data and predicts the charge for the new persons that the medical insurance should cover. The data is to be divided to 1000 samples for both training and validation and the rest is for testing. The dataset can be downloaded from:

https://www.kaggle.com/mirichoi0218/insurance

Project steps are as follows:

1. Load the dataset. **(Phase 1)**
2. Prepare the train-validation and test portions. **(Phase 1)**
3. Apply any preprocessing or features that you find suitable for the data. **(Phase 2)**
4. Apply 3 different models and compare between them, 2 mandatory and the 3rd is bonus. **(Phase 3)**

Phase-1 load data set

```
########################################################  Phase -1  ##########################################################

import matplotlib.pyplot as plt
import numpy as np
import pandas as pd

############################################ 1. LOAD DATA SET ##################################################################
path = 'F:\\College\\AI_year_3\\Machine Learning\\Assigments\\Project\\archive\\insurance.csv'
df = pd.read_csv(path)
X = df.iloc[:, :-1].values
y = df.iloc[:, -1].values
print('-----------------------------------------------------------------------------------------------------------------')
print('X IS :')
print(X)
print('-----------------------------------------------------------------------------------------------------------------')
print('Y IS :')
print(y)
print('-----------------------------------------------------------------------------------------------------------------')
```

Phase-2 Processing data , Feature scalinig data

```
################################################ Encoding ############################################################
from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import OneHotEncoder
ct = ColumnTransformer(
    [('one_hot_encoder', OneHotEncoder(categories='auto'), [1, 4, 5])],
    remainder='passthrough'
)
X = ct.fit_transform(X)
# One hot encode the data

################################################ Feature Scaling #######################################################
from sklearn.preprocessing import StandardScaler
ss = StandardScaler()
X = ss.fit_transform(X)

```

Train-test Split

```
############################################## Train-test Split #######################################################
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.20, random_state=0, shuffle=True)

```

Phase-3 Applying different models

1-# ### Multiple linear regression

```
# ### Multiple linear regression
from sklearn.linear_model import LinearRegression
linear_reg_model1 = LinearRegression()
linear_reg_model1.fit(X_train, y_train)
y_pred_LR = linear_reg_model1.predict(X_test)
np.set_printoptions(precision=2)
print(np.concatenate((y_pred_LR.reshape(len(y_pred_LR), 1),
                      y_test.reshape(len(y_test), 1)), 1)[:10, :])
print('-----------------------------------------------------------------------------------------------------------')
```

2- # ### Polynomial regression

```
# ### Polynomial regression
from sklearn.preprocessing import PolynomialFeatures
poly_reg = PolynomialFeatures(degree=2)
poly_reg_model2 = LinearRegression()
poly_reg_model2.fit(poly_reg.fit_transform(X_train), y_train)
y_pred_PR = poly_reg_model2.predict(poly_reg.fit_transform(X_test))
print(np.concatenate((y_pred_PR.reshape(len(y_pred_PR), 1),
                      y_test.reshape(len(y_test), 1)), 1)[:10, :])
print('-----------------------------------------------------------------------------------------------------------')
```

3- # ### Decision Tree Regression

```
# ### Decision Tree Regression
from sklearn.tree import DecisionTreeRegressor
decisionTree_reg_model3 = DecisionTreeRegressor(random_state=0)
decisionTree_reg_model3.fit(X_train, y_train)
y_pred_DTR = decisionTree_reg_model3.predict(X_test)
print(np.concatenate((y_pred_DTR.reshape(len(y_pred_DTR), 1),
                      y_test.reshape(len(y_test), 1)), 1)[:10, :])
print('-------------------------------------------------------------------------------------------------------')
```

4-# ### Random Forest Regression

```
from sklearn.ensemble import RandomForestRegressor
randoForest_reg_model4 = RandomForestRegressor(n_estimators=300, random_state=0)
randoForest_reg_model4.fit(X_train, y_train)
y_pred_RFR = randoForest_reg_model4.predict(X_test)
print(np.concatenate((y_pred_RFR.reshape(len(y_pred_RFR), 1),
                      y_test.reshape(len(y_test), 1)), 1)[:10, :])
print('-------------------------------------------------------------------------------------------------------')
```

5-# ### Support Vector Regression

```
from sklearn.svm import SVR
supportVector_reg_model5 = SVR(kernel='linear', C=20, epsilon=0.01)
supportVector_reg_model5.fit(X_train, y_train)
y_pred_SVR = supportVector_reg_model5.predict(X_test)
print(np.concatenate((y_pred_SVR.reshape(len(y_pred_SVR), 1),
                      y_test.reshape(len(y_test), 1)), 1)[:10, :])
print('-------------------------------------------------------------------------------------------------------')
```

```
# ## Evaluating Performance
from sklearn.metrics import r2_score
print('Evaluating Performance:-')

# ### Multiple Linear Regression
print('Multiple Linear Regression:', r2_score(y_test, y_pred_LR))

# ### Polynomial Regression
print('Polynomial Regression:', r2_score(y_test, y_pred_PR))

# ### Decision Tree Regression
print('Decision Tree Regression:', r2_score(y_test, y_pred_DTR))

# ### Random Forest Regression
print('Random Forest Regression:', r2_score(y_test, y_pred_RFR))

# ### Support Vector Regression
print('Support Vector Regression:', r2_score(y_test, y_pred_SVR))

print('-------------------------------------------------------------------------------------------------------')
```

Output:-
C:\Users\hecen\AppData\Local\Programs\Python\Python39\python.exe "F:/College/AI_year_3/Machine Learning/Assigments/Project/Code/main.py"

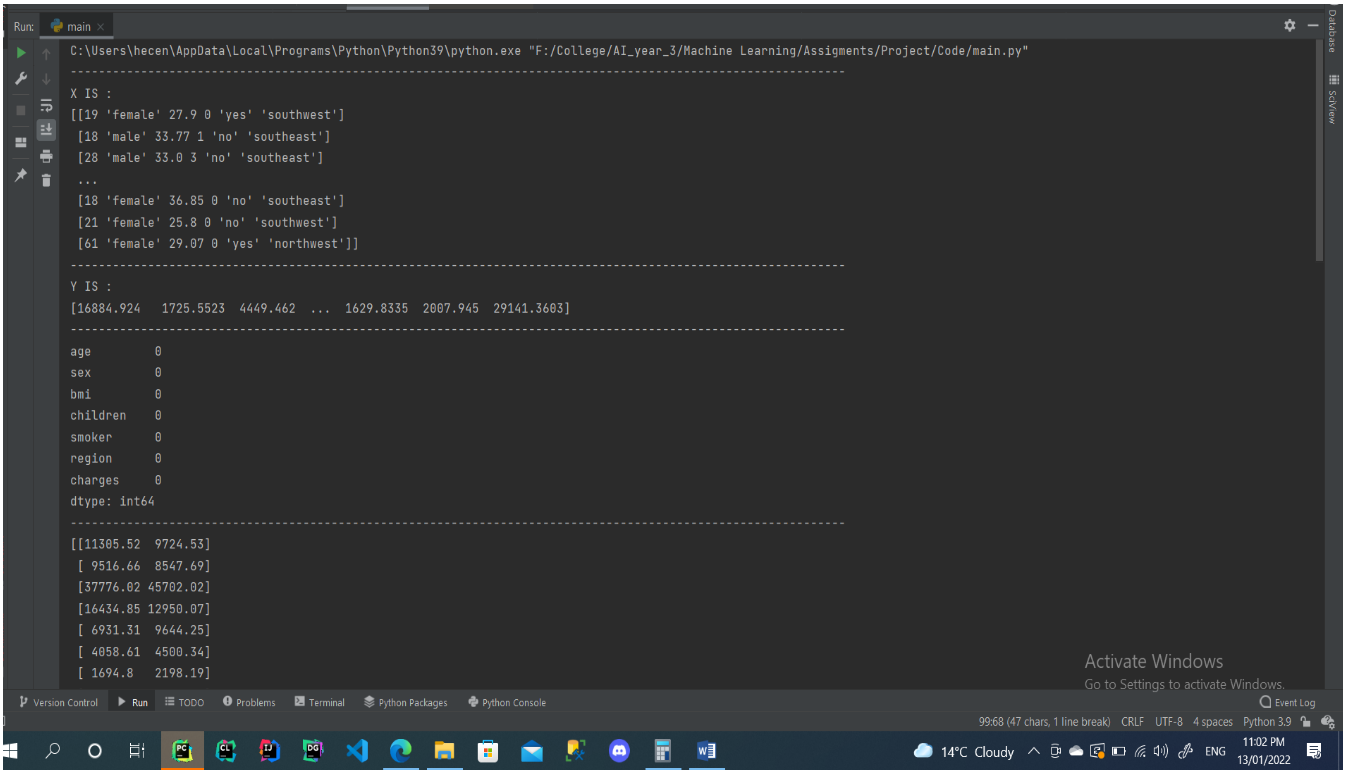
```
----------------------------------------------------------------------------------------------------------
X IS :
[[19 'female' 27.9 0 'yes' 'southwest']
 [18 'male' 33.77 1 'no' 'southeast']
 [28 'male' 33.0 3 'no' 'southeast']
 ...
 [18 'female' 36.85 0 'no' 'southeast']
 [21 'female' 25.8 0 'no' 'southwest']
 [61 'female' 29.07 0 'yes' 'northwest']]
----------------------------------------------------------------------------------------------------------
Y IS :
[16884.924   1725.5523  4449.462  ...  1629.8335  2007.945
29141.3603]
----------------------------------------------------------------------------------------------------------
age         0
sex         0
bmi         0
children    0
smoker      0
region      0
charges     0
dtype: int64
----------------------------------------------------------------------------------------------------------
```

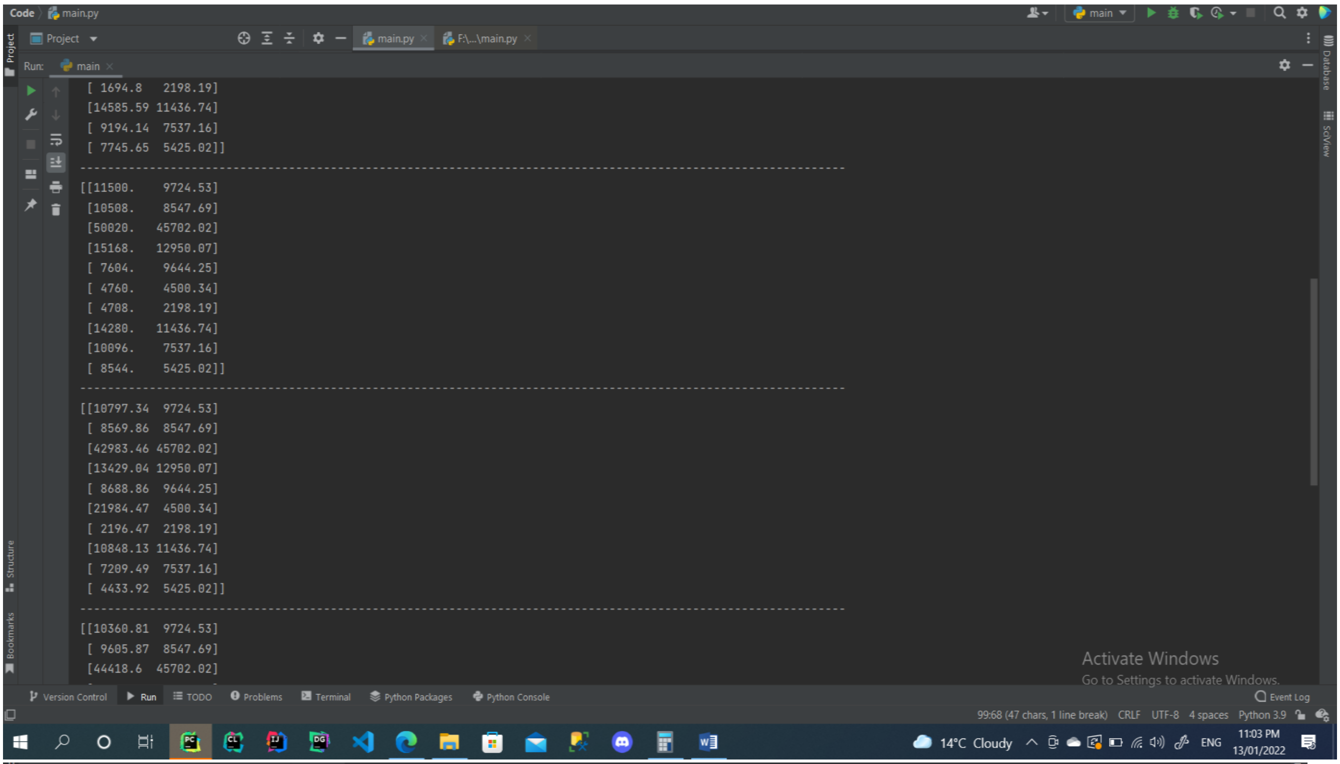
```
[[11305.52  9724.53]
 [ 9516.66  8547.69]
 [37776.02 45702.02]
 [16434.85 12950.07]
 [ 6931.31  9644.25]
 [ 4058.61  4500.34]
 [ 1694.8   2198.19]
 [14585.59 11436.74]
 [ 9194.14  7537.16]
 [ 7745.65  5425.02]]
-------------------------------------------------------------------------------------------------------------------
[[11500.    9724.53]
 [10508.    8547.69]
 [50020.   45702.02]
 [15168.   12950.07]
 [ 7604.    9644.25]
 [ 4760.    4500.34]
 [ 4708.    2198.19]
 [14280.   11436.74]
 [10096.    7537.16]
 [ 8544.    5425.02]]
-------------------------------------------------------------------------------------------------------------------
[[10797.34  9724.53]
 [ 8569.86  8547.69]
 [42983.46 45702.02]
 [13429.04 12950.07]
 [ 8688.86  9644.25]
 [21984.47  4500.34]
 [ 2196.47  2198.19]
```

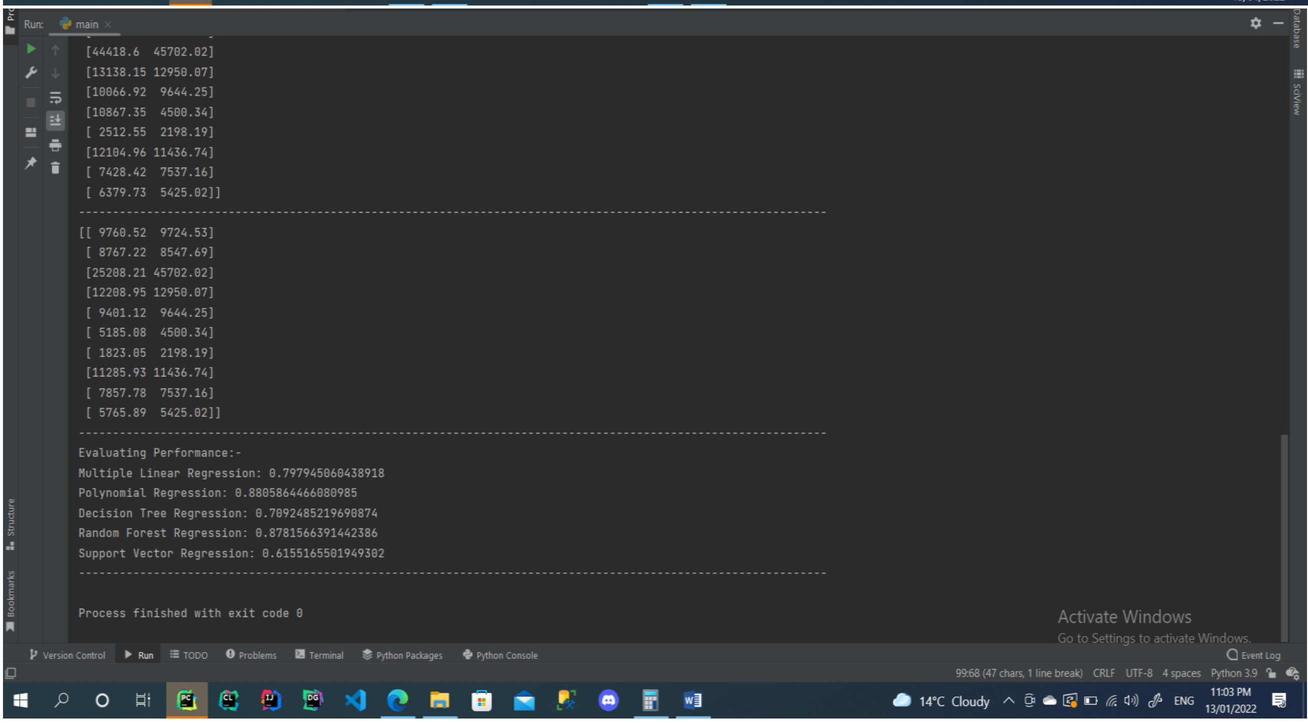
```
 [10848.13 11436.74]
 [ 7209.49  7537.16]
 [ 4433.92  5425.02]]
-----------------------------------------------------------------------------------------------------------------
[[10360.81  9724.53]
 [ 9605.87  8547.69]
 [44418.6  45702.02]
 [13138.15 12950.07]
 [10066.92  9644.25]
 [10867.35  4500.34]
 [ 2512.55  2198.19]
 [12104.96 11436.74]
 [ 7428.42  7537.16]
 [ 6379.73  5425.02]]
-----------------------------------------------------------------------------------------------------------------
[[ 9760.52  9724.53]
 [ 8767.22  8547.69]
 [25208.21 45702.02]
 [12208.95 12950.07]
 [ 9401.12  9644.25]
 [ 5185.08  4500.34]
 [ 1823.05  2198.19]
 [11285.93 11436.74]
 [ 7857.78  7537.16]
 [ 5765.89  5425.02]]
-----------------------------------------------------------------------------------------------------------------
Evaluating Performance:-
Multiple Linear Regression: 0.797945060438918
```

Polynomial Regression: 0.8805864466080985
Decision Tree Regression: 0.7092485219690874
Random Forest Regression: 0.8781566391442386
Support Vector Regression: 0.6155165501949302

Process finished with exit code 0







Our Comment

Model	R square
Multiple Linear Regression:	0.797945060438918
Polynomial Regression:	0.8805864466080985
Decision Tree Regression:	0.7092485219690874
Random Forest Regression:	0.8781566391442386

Support Vector Regression: 0.6155165501949302

From the result above, it is clear that for the present problem, the best performing model is Polynomial Regression with the highest R square (Coefficient of Determination). But we have to keep in mind that Random Forest Regression is also not far behind with respect to the metrics.

**suggested improvements should also be included:
By decreasing test size to 10% and increase train size to 90% it would help to increase the the performance exept SVM model ……………………….

```
Evaluating Performance:-
Multiple Linear Regression: 0.8260991689905677
Polynomial Regression: 0.9074506895098615
Decision Tree Regression: 0.7751071099718194
Random Forest Regression: 0.8986104800375482
Support Vector Regression: 0.586355773400276
```

If the data set is small and we need a good prediction for the response variable , it is a good idea to go for models like Random Forest or Decision tree. As they are capable of generating good prediction with lesser training data or labelled data.

```python
#####################################################  Phase -1
#######################################################

import matplotlib.pyplot as plt
import numpy as np
import pandas as pd

########################################### 1. LOAD DATA SET
###########################################################
path = 'F:\\College\\AI_year_3\\Machine
Learning\\Assigments\\Project\\archive\\insurance.csv'
df = pd.read_csv(path)
X = df.iloc[:, :-1].values
y = df.iloc[:, -1].values
print('--------------------------------------------------------------------------------
------------------------------')
print('X IS :')
print(X)
print('--------------------------------------------------------------------------------
------------------------------')
print('Y IS :')
print(y)
print('--------------------------------------------------------------------------------
------------------------------')


#####################################################  Phase -2
#######################################################

############################################### Handle Empty cells
#####################################################
# Check if the data frame contain null values
print(df.isnull().sum())
# Check completed df contains no null values
print('--------------------------------------------------------------------------------
------------------------------')

################################################# Encoding
#############################################################
from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import OneHotEncoder
ct = ColumnTransformer(
    [('one_hot_encoder', OneHotEncoder(categories='auto'), [1, 4, 5])],
    remainder='passthrough'
)
X = ct.fit_transform(X)
# One hot encode the data

################################################# Feature Scaling
######################################################
from sklearn.preprocessing import StandardScaler
ss = StandardScaler()
X = ss.fit_transform(X)


############################################### Train-test Split
#######################################################
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=0, shuffle=True)

#####################################################  Phase -3
#######################################################
################################################ Applying Models
#######################################################
```

```python
# ### Multiple linear regression
from sklearn.linear_model import LinearRegression
linear_reg_model1 = LinearRegression()
linear_reg_model1.fit(X_train, y_train)
y_pred_LR = linear_reg_model1.predict(X_test)
np.set_printoptions(precision=2)
print(np.concatenate((y_pred_LR.reshape(len(y_pred_LR), 1),
                      y_test.reshape(len(y_test), 1)), 1)[:10, :])
print('--------------------------------------------------------------------------------
------------------------------')


# ### Polynomial regression
from sklearn.preprocessing import PolynomialFeatures
poly_reg = PolynomialFeatures(degree=2)
poly_reg_model2 = LinearRegression()
poly_reg_model2.fit(poly_reg.fit_transform(X_train), y_train)
y_pred_PR = poly_reg_model2.predict(poly_reg.fit_transform(X_test))
print(np.concatenate((y_pred_PR.reshape(len(y_pred_PR), 1),
                      y_test.reshape(len(y_test), 1)), 1)[:10, :])
print('--------------------------------------------------------------------------------
------------------------------')


# ### Decision Tree Regression
from sklearn.tree import DecisionTreeRegressor
decisionTree_reg_model3 = DecisionTreeRegressor(random_state=0)
decisionTree_reg_model3.fit(X_train, y_train)
y_pred_DTR = decisionTree_reg_model3.predict(X_test)
print(np.concatenate((y_pred_DTR.reshape(len(y_pred_DTR), 1),
                      y_test.reshape(len(y_test), 1)), 1)[:10, :])
print('--------------------------------------------------------------------------------
------------------------------')


# ### Random Forest Regression
from sklearn.ensemble import RandomForestRegressor
randoForest_reg_model4 = RandomForestRegressor(n_estimators=300, random_state=0)
randoForest_reg_model4.fit(X_train, y_train)
y_pred_RFR = randoForest_reg_model4.predict(X_test)
print(np.concatenate((y_pred_RFR.reshape(len(y_pred_RFR), 1),
                      y_test.reshape(len(y_test), 1)), 1)[:10, :])
print('--------------------------------------------------------------------------------
------------------------------')


# ### Support Vector Regression
from sklearn.svm import SVR
supportVector_reg_model5 = SVR(kernel='linear', C=20, epsilon=0.01)
supportVector_reg_model5.fit(X_train, y_train)
y_pred_SVR = supportVector_reg_model5.predict(X_test)
print(np.concatenate((y_pred_SVR.reshape(len(y_pred_SVR), 1),
                      y_test.reshape(len(y_test), 1)), 1)[:10, :])
print('--------------------------------------------------------------------------------
------------------------------')


# ## Evaluating Performance
from sklearn.metrics import r2_score
print('Evaluating Performance:-')

# ### Multiple Linear Regression
print('Multiple Linear Regression:', r2_score(y_test, y_pred_LR))

# ### Polynomial Regression
print('Polynomial Regression:', r2_score(y_test, y_pred_PR))
```

```
# ### Decision Tree Regression
print('Decision Tree Regression:', r2_score(y_test, y_pred_DTR))

# ### Random Forest Regression
print('Random Forest Regression:', r2_score(y_test, y_pred_RFR))

# ### Support Vector Regression
print('Support Vector Regression:', r2_score(y_test, y_pred_SVR))

print('------------------------------------------------------------------------------
-----------------------------')
```