

REGRESSION ASSIGNMENT

Problem Statement: To predict the insurance charges for clients

1) Identifying the problem statement:

- a) Stage 1: Machine Learning
- b) Stage 2: Supervised Learning
- c) Stage 3: Regression

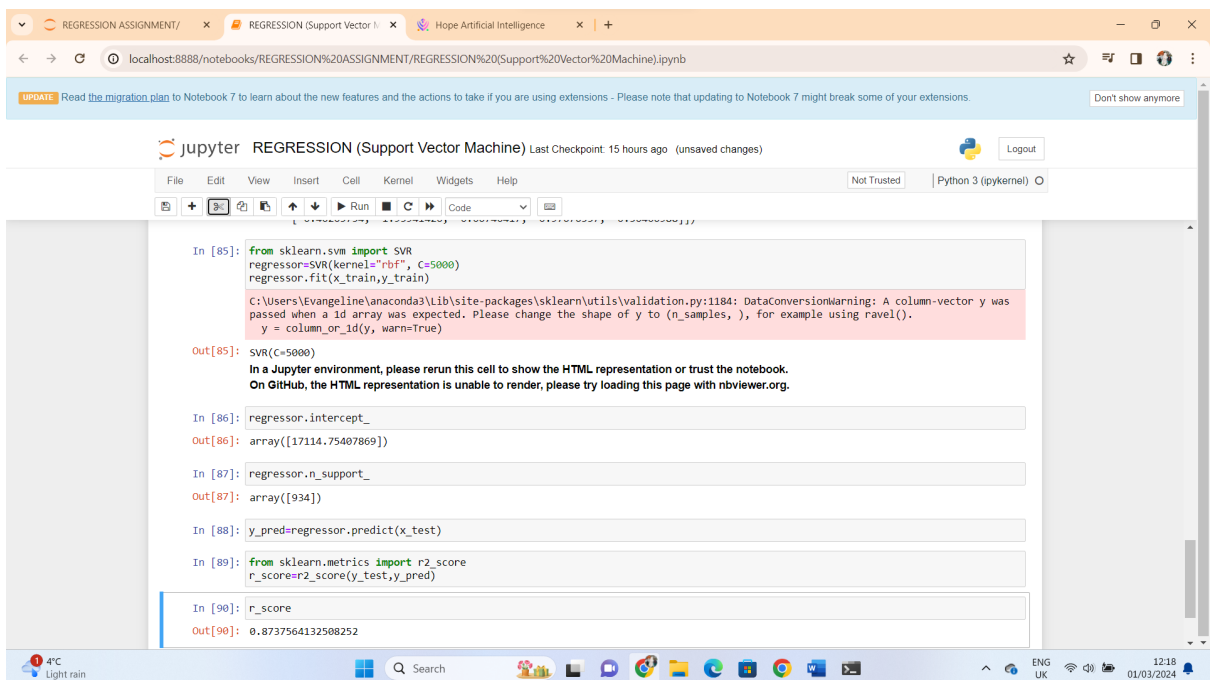
2) Information about the Dataset:

There are 1338 Rows and 6 Columns in the given Insurance Dataset. Inputs and Output present.

3) Mentioning the pre-processing method:

I have used One-Hot-encoding to convert categorical data to numerical data, this is known as Nominal Data. A Nominal Data is used to label variables without providing any quantitative Value.

4) Developing a good model: Support Vector Machine has given me a r^2 Value of 0.87375



```
In [85]: from sklearn.svm import SVR
regressor=SVR(kernel="rbf", C=5000)
regressor.fit(x_train,y_train)

C:\Users\Evangeline\anaconda3\Lib\site-packages\sklearn\utils\validation.py:1184: DataConversionWarning: A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n_samples, ), for example using ravel().
y = column_or_1d(y, warn=True)

Out[85]: SVR(C=5000)

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [86]: regressor.intercept_
Out[86]: array([17114.75407869])

In [87]: regressor.n_support_
Out[87]: array([934])

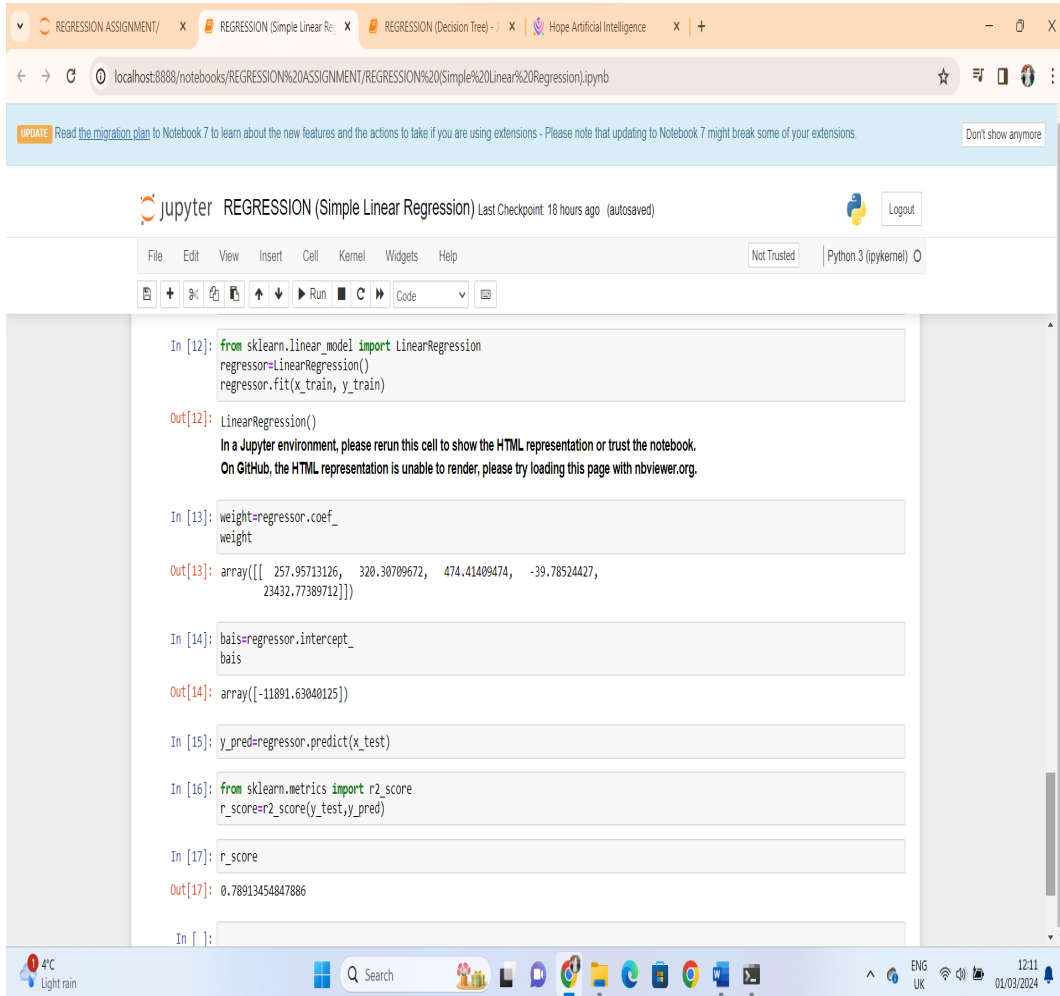
In [88]: y_pred=regressor.predict(x_test)

In [89]: from sklearn.metrics import r2_score
r_score=r2_score(y_test,y_pred)

In [90]: r_score
Out[90]: 0.8737564132508252
```

5)Regression Model research R2 Values:

a) Simple Linear Regression R2 Value=0.78913



The screenshot shows a Jupyter Notebook interface with the title 'REGRESSION (Simple Linear Regression)'. The notebook contains several code cells and their outputs. The first cell imports `LinearRegression` from `sklearn.linear_model` and fits the model to training data. The second cell prints the `coef_` attribute, showing an array of coefficients. The third cell prints the `intercept_` attribute, showing the intercept value. The fourth cell prints the `predict` method results for test data. The fifth cell imports `r2_score` from `sklearn.metrics` and calculates the R-squared value. The final output shows the R-squared value as 0.78913454847886.

```
In [12]: from sklearn.linear_model import LinearRegression
regressor=LinearRegression()
regressor.fit(x_train, y_train)

Out[12]: LinearRegression()
In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [13]: weight=regressor.coef_
weight

Out[13]: array([[ 257.95713126,  320.30709672,  474.41409474, -39.78524427,
                  23432.77389712]])

In [14]: bais=regressor.intercept_
bais

Out[14]: array([-11891.63040125])

In [15]: y_pred=regressor.predict(x_test)

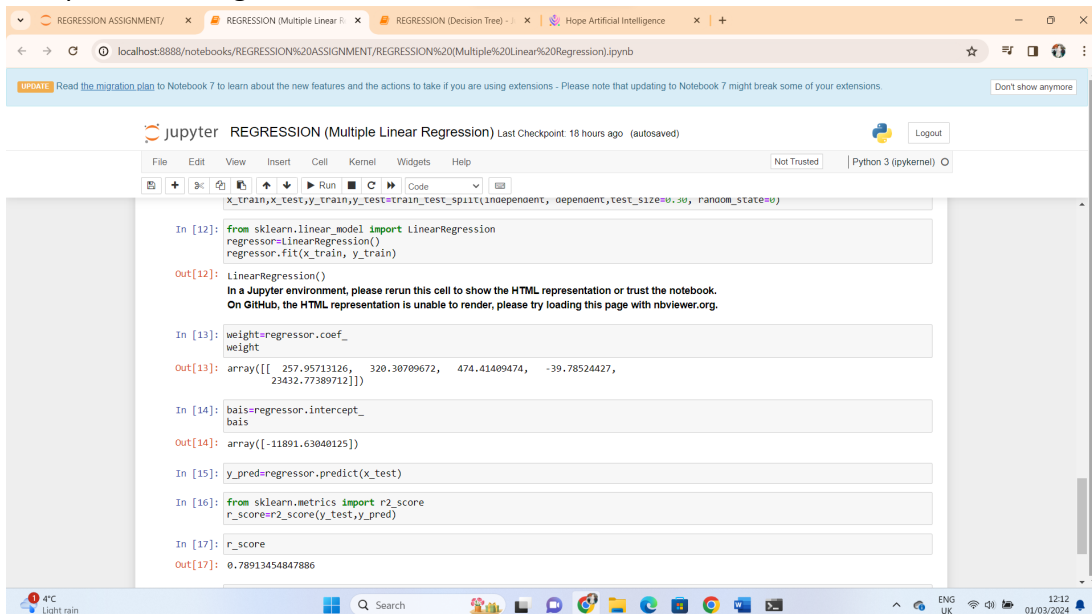
In [16]: from sklearn.metrics import r2_score
r_score=r2_score(y_test,y_pred)

In [17]: r_score

Out[17]: 0.78913454847886

In [ ]:
```

b) Multiple Linear Regression R2 Value=0.78913



The screenshot shows a Jupyter Notebook interface with the title 'REGRESSION (Multiple Linear Regression)'. The notebook contains several code cells and their outputs. The first cell imports `LinearRegression` from `sklearn.linear_model` and fits the model to training data. The second cell prints the `coef_` attribute, showing an array of coefficients. The third cell prints the `intercept_` attribute, showing the intercept value. The fourth cell prints the `predict` method results for test data. The fifth cell imports `r2_score` from `sklearn.metrics` and calculates the R-squared value. The final output shows the R-squared value as 0.78913454847886.

```
x_train,x_test,y_train,y_test=train_test_split(independent, oependent, test_size=0.30, random_state=0)

In [12]: from sklearn.linear_model import LinearRegression
regressor=LinearRegression()
regressor.fit(x_train, y_train)

Out[12]: LinearRegression()
In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [13]: weight=regressor.coef_
weight

Out[13]: array([[ 257.95713126,  320.30709672,  474.41409474, -39.78524427,
                  23432.77389712]])

In [14]: bais=regressor.intercept_
bais

Out[14]: array([-11891.63040125])

In [15]: y_pred=regressor.predict(x_test)

In [16]: from sklearn.metrics import r2_score
r_score=r2_score(y_test,y_pred)

In [17]: r_score

Out[17]: 0.78913454847886
```

c) Decision Tree R2 Value = 0.7467710

The screenshot shows a Jupyter Notebook titled 'REGRESSION (Decision Tree)' with the following code and output:

```

In [9]: from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(independent, dependent,test_size=0.30, random_state=0)

In [43]: from sklearn.tree import DecisionTreeRegressor
regressor=DecisionTreeRegressor(min_samples_leaf=1, splitter='random')
regressor=regressor.fit(x_train,y_train)

In [44]: y_pred=regressor.predict(x_test)

In [45]: from sklearn.metrics import r2_score
r_score=r2_score(y_test,y_pred)

In [46]: r_score
Out[46]: 0.7467710817251414

```

3) Decision Tree Regression

No	CRITERION	SPLITTER	R2 VALUE
1.	Squared_Error	Best	0.6781754
2.	Squared Error	Random	0.6927514
3.	Friedman_mse	Best	0.6794144
4.	Friedman_mse	Random	0.6931777
5.	Absolute_error	Best	0.7345476
6.	Absolute_error	Random	0.6780406
7.	Poisson	Best	0.6836392
8.	Poisson	Random	0.7259208
9.	min_samples_split=2	Random	0.7361561
10.	min_samples_split=2	Best	0.6891230
11.	min_samples_leaf=1	Best	0.6859465

12.	min_samples_leaf=1	Random	0.7467710
13.	random_state=None	Random	0.7128355
14.	random_state=None	Best	0.6571168
15.	max_leaf_nodes=None	Best	0.6637487
16.	max_leaf_nodes=None	Random	0.7185674

The best R2 Value for Decision Tree Regression is min_samples_leaf=1 and Random hyper tuning Parameter = 0.7467710

SUPPORT VECTOR MACHINE:

a) Support Vector Machine R2 Value (Kernel ='RBF', C=5000) = 0.8737564

No	Hyper Parameter	Linear (r value)	Poly (r value)	RBF (Non-Linear) (r value)	Sigmoid (r value)
1.	C=10	0.46242633	0.0386251	-0.0323806	-0.098553
2.	C=100	0.6289632	0.6164698	0.3196645	0.5268415
3.	C=1000	0.7648394	0.8546515	0.8107195	0.212045
4.	C=2000	0.743935	0.8583433	0.85407306	-0.621621
5.	C=3000	0.7413370	0.8580853	0.8646256	-2.143154
6	C=5000	0.7413297	0.8587213	0.8737564	-8.160605

```

In [85]: from sklearn.svm import SVR
regressor=SVR(kernel="rbf", C=5000)
regressor.fit(x_train,y_train)

C:\Users\Evangeline\anaconda3\lib\site-packages\sklearn\utils\validation.py:1184: DataConversionWarning: A column-vector y was
passed when a 1d array was expected. Please change the shape of y to (n_samples, ), for example using ravel().
y = column_or_1d(y, warn=True)

Out[85]: SVR(C=5000)
In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [86]: regressor.intercept_
Out[86]: array([17114.75407869])

In [87]: regressor.n_support_
Out[87]: array([934])

In [88]: y_pred=regressor.predict(x_test)

In [89]: from sklearn.metrics import r2_score
r_score=r2_score(y_test,y_pred)

In [90]: r_score
Out[90]: 0.8737564132508252

```

RANDOM FOREST

Random Forest (**n_estimators=100, Max_Feature='log2'**) R2 Value = **0.866709**

NO	PARAMETER n_estimators	Hyper Tuning Parameters	R2 Value
1.	50	criterion='squared_error	0.849944
2.	100	criterion='squared_error	0.851487
3.	50	Absolute_error	0.851068
4.	100	Absolute_error	0.857766
5.	50	Friedman_mse	0.847555
6.	100	Friedman_mse	0.852700
7.	50	Poisson	0.854152
8.	100	Poisson	0.854285
9.	50	Max_Features=sqrt	0.858814
10.	100	Max_Features=sqrt	0.864422

11.	50	Max_Features=Log2	0.864218
12.	100	Max_Features=Log2	0.866709
13.	50	Max_depth=None	0.848391
14.	100	Max_depth=None	0.849288
15.	50	Min_Samples_Split=2	0.847695
16.	100	Min_Samples_Split=2	0.851336
17.	50	Max_Features=1.0	0.848633
18.	100	Max_leaf_Nodes=None	0.850710

```

1336 2007
1337 29141
1338 rows x 1 columns

In [9]: from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(independent, dependent, test_size=0.30, random_state=0)

In [30]: from sklearn.ensemble import RandomForestRegressor
regressor=RandomForestRegressor(n_estimators=100, max_features='log2')
regressor.fit(x_train,y_train)

C:\Users\Evangelina\anaconda3\lib\site-packages\sklearn\base.py:1151: DataConversionWarning: A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n_samples,), for example using ravel().
return fit_method(estimator, *args, **kwargs)

Out[30]:
RandomForestRegressor
RandomForestRegressor(max_features='log2')

In [31]: y_pred=regressor.predict(x_test)

In [32]: from sklearn.metrics import r2_score
r_score=r2_score(y_test,y_pred)

In [33]: r_score
Out[33]: 0.8667066765752847

```

6) I have chosen **Support Vector Machine** as my final model because SVM algorithm has given me the best **R2 Value (Kernel ='RBF', C=5000) = 0.8737564** compared to the other regression algorithm.

