Assignment-Regression Algorithm

Insurance Charges

1.) Identify your problem statement

Ans:To predict the insurance charges based on multiple input parameters provided by the client.

2.) Tell basic info about the dataset (Total number of rows, columns)

Ans:

Total Rows: 1338
Total Columns: 6
Input Features (X): 5
Target Variable (Y): `charges` (Insurance Charges)

3.) Mention the pre-processing method if you're doing any (like converting string to number – nominal data)

Ans:

- `sex` and `smoker` are categorical variables that need to be converted into numerical values.
- Used One-Hot Encoding (`pd.get_dummies`) to transform these categorical variables.

A	B	C	D	E	F
age	sex	bmi	children	smoker	charges
19	female	27.9	0	yes	16884.924
18	male	33.77	1	no	1725.5523
28	male	33	3	no	4449.462
33	male	22.705	0	no	21984.47061

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.

Ans:

final model: Random Forest, based on the other Algorithm R2 score value

```
#running for the highest score 0.89
regressor = RandomForestRegressor(n_estimators=200,  criterion= 'absolute_error', ma
regressor = regressor.fit(X_train, Y_train)
#test using the test data
y_pred = regressor.predict(X_test)
# evalulate metrics R2 score
from sklearn.metrics import r2_score
r_score = r2_score(Y_test, y_pred)

print("r2_score for the 0.89", r_score)
print("----------------------------")

import pickle
filename = "finalized_InsuranceChargePrediction_RF_Model.sav"
```

To conclude before Random Forest model was the final model with prediction percentage of 89% below were the Algorithms tried and tuned to get the top results

Assignment-Regression Algorithm

Insurance Charges

Algorithm 1: Multiple Linear Regression: **0.78** is the highest R2 Score

```
##################1.Multiple Linear Regression #################
#possible Hyper tuning params fit_intercept=True, copy_X=True, n_jobs=None, positive=False
from sklearn.linear_model import LinearRegression
def calculateModelR2Score_MLR(fit_intercept_V, copy_X_V, n_jobs_V, positive_V):
    regressor = LinearRegression(fit_intercept=fit_intercept_V, copy_X=copy_X_V, n_jobs=n_jobs_V, positive=positive_V)
    regressor = regressor.fit(X_train, Y_train)

    #test using the test data
    y_pred = regressor.predict(X_test)

    # evalulate metrics R2 score
    from sklearn.metrics import r2_score
    r_score = r2_score(Y_test, y_pred)
    print("fit_intercept:", fit_intercept_V)
    print("copy_X:", copy_X_V)
    print("n_jobs:", n_jobs_V)
    print("positive:", positive_V)
    print("r2_score", r_score)
    print("----------------------------")

fit_intercept_List = [True, False]
copy_X_List = [False, True]
n_jobs_List = [0] #,1,2]
positive_List = [False, True]
```

Sorted Multiple Linear Regression R2 Score Table

	fit_intercept	copy_X	n_jobs	positive	r2_score
1	True	False	0	False	0.7894790349867009
2	True	True	0	False	0.7894790349867009
3	True	False	0	True	0.7894429387120754
4	True	True	0	True	0.7894429387120754
5	False	False	0	False	0.7623723282279132
6	False	True	0	False	0.7623723282279132
7	False	False	0	True	0.7614332936145434
8	False	True	0	True	0.7614332936145434

Algorithm 2: Scalar Vector Model SVM: **00.85** is the highest R2 Score

Assignment-Regression Algorithm

Insurance Charges

```
#################2.Scalar Vector Model (SVM) ################
#possible Hyper tuning params: sklearn.svm.SVR(*, kernel='rbf', degree=3, gamma='scale', coef0=0.0, tol=0.001, C=1.0, epsilon=0
from sklearn.svm import SVR
def calculateModelR2Score_SVM(kernel_V, degree_V, gamma_V, C_V, epsilon_V):
    regressor = SVR(kernel=kernel_V, degree=degree_V, gamma=gamma_V, C=C_V, epsilon=epsilon_V )
    regressor = regressor.fit(X_train, Y_train)

    #test using the test data
    y_pred = regressor.predict(X_test)

    # evalulate metrics R2 score
    from sklearn.metrics import r2_score
    r_score = r2_score(Y_test, y_pred)

    print("kernel_V", kernel_V)
    print("degree_V", degree_V)
    print("gamma_V ", gamma_V)
    print("C_V ", C_V)
    print("epsilon_V ", epsilon_V)
    print("r2_score", r_score)
    print("----------------------------")

kernel_List= ['rbf', 'linear', 'poly']
C_List= [1] #0.1, 1, 10, 100]
gamma_List= [1] #['scale', 'auto', 0.01, 0.1, 1]
epsilon_List= [0.001] #, 0.01, 0.1, 1.0]
degree_List= [2] #, 3, 4]
```

Sorted SVM R2 Score Table

	kernel	degree	gamma	C	epsilon	r2_score
1	poly	2	1	1	0.001	0.8538275261915 09
2	rbf	2	1	1	0.001	-0.089676683997 926
3	linear	2	1	1	0.001	-0.111671877542 268

Algorithm 3: Decision Tree Regressor - **00.88** is the highest R2 Score

```
#possible Hyper tuning params: DecisionTreeRegressor(*, criterion='squared_error', splitter='best', max_depth=None, min_samples_split
from sklearn.tree import DecisionTreeRegressor
def calculateModelR2Score_DecisionTree(criterion_V, splitter_V, max_depth_V, min_samples_split_V, min_samples_leaf_V):
    regressor = DecisionTreeRegressor(criterion=criterion_V, splitter=splitter_V, max_depth=max_depth_V, min_samples_split=min_sample

    regressor = regressor.fit(X_train, Y_train)

    #test using the test data
    y_pred = regressor.predict(X_test)

    # evalulate metrics R2 score
    from sklearn.metrics import r2_score
    r_score = r2_score(Y_test, y_pred)
    if(r_score > 0.886):
        print("criterion_V", criterion_V)
        print("splitter_V", splitter_V)
        print("max_depth_V ", max_depth_V)
        print("min_samples_split_V ", min_samples_split_V)
        print("min_samples_leaf_V ", min_samples_leaf_V)
        print("r2_score", r_score)
        print("----------------------------")



criterion_List = ["squared_error", "friedman_mse", "absolute_error", "poisson"]
splitter_List = ["best", "random"]
max_depth_List = [None, 3, 5, 10, 20]
min_samples_split_List = [2, 5, 10, 20, 50]
min_samples_leaf_List = [1, 5, 10, 20, 50]
# min_weight_fraction_leaf_List = [0.0, 0.01, 0.05, 0.1]
```

Assignment-Regression Algorithm

Insurance Charges

Sorted Decision Tree R2 Score Table

	criterion	splitter	max_depth	min_samples_split	min_samples_leaf	r2_score
1	absolute_error	best		20	5	0.8863459882523649
2	absolute_error	best	10.0	20	5	0.8863459882523649
3	absolute_error	best	20.0	20	5	0.8863459882523649
4	absolute_error	best	5.0	2	5	0.8860838477714783
5	absolute_error	best	5.0	5	5	0.8860838477714783
6	absolute_error	best	5.0	10	5	0.8860838477714783

Algorithm 4: Random Forest Regressor - **00.89** is the highest R2 Score

```
##################4.Random Forest #################
#possible Hyper tuning params: class sklearn.ensemble.RandomForestRegressor(n_estimators=100,
from sklearn.ensemble import RandomForestRegressor
def calculateModelR2Score_RF(n_estimators_Value, criterion_value, max_depth_Value, min_sample
    regressor = RandomForestRegressor(n_estimators=n_estimators_Value, criterion=criterion_va
    regressor = regressor.fit(X_train, Y_train)
    #test using the test data
    y_pred = regressor.predict(X_test)
    # evalulate metrics R2 score
    from sklearn.metrics import r2_score
    r_score = r2_score(Y_test, y_pred)
    if(r_score > 0.891):
        print("n_estimators_Value", n_estimators_Value)
        print("criterion_Value", criterion_value)
        print("max_depth_Value", max_depth_Value)
        print("min_samples_split_Value", min_samples_split_Value)
        print("min_samples_leaf_Value", min_samples_leaf_Value)
        print("max_features_Value", max_features_Value)
        print("bootstrap_Value", bootstrap_Value)
        print("oob_score_Value", oob_score_Value)
        print("random_state_Value", random_state_Value)
        print("r2_score", r_score)
        print("----------------------------")
n_estimators_List = [200] #100, 200] #, 300, 500]  # Number of trees
criterion_List = ['absolute_error']#'squared_error', 'absolute_error', 'friedman_mse']
max_depth_List = [None, 10] #, 20, 30, 50]  # Maximum depth of trees
min_samples_split_List = [2, 5] #, 10]  # Minimum samples required to split a node
min_samples_leaf_List = [1, 2, 4]#, 10]  # Minimum samples required at each leaf node
max_features_List = ['sqrt', 'log2', None]  # Number of features to consider at each split
bootstrap_List = [True] #, False]  # Whether bootstrap samples are used
oob_score_List = [True, False]  # Whether to use out-of-bag samples for scoring
random_state_List = [30, 42, 50]  # For reproducibility
```

Assignment-Regression Algorithm

Insurance Charges

n_estimators	criterion	max_depth	min_samples_split	min_samples_leaf	max_features	bootstrap	oob_score	random_state	r2_score
200	absolute_error	10.0	2	4		True	True	42	0.8914717067507086
200	absolute_error	10.0	2	4		True	False	42	0.8914717067507086
200	absolute_error	10.0	5	4		True	True	42	0.8914717067507086
200	absolute_error	10.0	5	4		True	False	42	0.8914717067507086
200	absolute_error		2	4		True	True	42	0.8913002216153745
200	absolute_error		2	4		True	False	42	0.8913002216153745
200	absolute_error		5	4		True	True	42	0.8913002216153745

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

Model	R^2 Score
Multiple Linear Regression	0.78
Support Vector Machine (SVM)	0.72
Decision Tree Regressor	0.88
Random Forest Regressor	**0.89**

6) Mention your final model, justify why u have chosen the same.
Ans: Selected Model: Random Forest Regressor

Justification:

- Achieved the highest R^2 score (0.89) among all models

Upload all the ipynb: Assignment/InsuranceCharges_ModelPrepare.ipynb at main · GeekPri/Assignment

final document in the pdf: Assignment/Assignment.pdf at main · GeekPri/Assignment