COMP4432 Machine Learning - Assignment 2

Learning objective: In Assignment 1, a linear regression model was constructed using the single best feature (identified from the correlations to the target variable) from the SciKit-Learn diabetes data set. Here, multiple algorithms will be considered that utilize the entire set of input features to identify an optimum model to estimate diabetes progression. In this assignment, will the following will be completed:

1. Data exploration. This is a reduced set of instructions similar to Assignment 1.
2. Initial model training. Three algorithms will be trained with the entire set of features from the diabetes data set. These three methods will utilize the default hyperparameters and the *cross_val_score* functionality.
3. Tuning of a decision tree. The optimal hyperparameter settings of a decision tree will be identified from examining hyperparameters that control depth.
4. Tuning of a random forest. By examining combination of hyperparameter settings, the optimal hyperparameters for a random forest will be identified.
5. From multiple candidate model, an optimum model will be selected.

Part 1: Data Exploration.

From Assignment 1, the diabetes data was shown to be well curated and clean (the data was nicely scaled, had no missing values, and had minimal skew). Examination of the dataset is redundant, so here only the correlation matrix will be examined again for later comparisons.

a) Load the scikit-learn diabetes bunch object into a variable.
b) Create a Pandas DataFrame from the bunch.
c) Construct a correlation matrix, and in a Markdown cell document the order of the correlations between the input features and the target variable.
d) Use a Seaborn pairplot to examine the scatter plots of the <u>four</u> features with the largest correlations to the target.
e) Partition the data into training and testing data sets using train_test_split.
    i) The testing data should be 20% of the original data set, and include all features in the DataFrame.
    ii) For reproducibility, set the random state. Use the same random state that you employed in Assignment Tip: You can produce the four data sets (X_train, y_train, X_test, y_test) in a single line of code.

Part 2: Model Training.

a) Instantiate the following three algorithms into appropriately named variables (do not fit the algorithms):
    i. linear regressor
    ii. decision tree regressor
    iii. random forest regressor (use a fixed random state with the random forest for reproducibility)
b) With the three algorithms instantiated above, and using *cross_val_score* with the *neg_root_mean_squared_error* scoring metric, *cv=5*, and the training data from Part 1e, calculate and print the following:
    i. The RMSE output from *cross_val_score* (should be an array with five elements).
    ii. The mean of the RMSE values
    iii. The standard deviation of the RMSE values
c) Let's gather the same information in Part 2b for the single feature linear regressor from Assignment 1. Using *cross_val_score* with the *neg_root_mean_squared_error* metric, and the training data from Part 1e reduced to only the single feature found in Assignment 1 (bmi), calculate and print the following:
    i. The RMSE output from *cross_val_score* (should be an array with five elements).
    ii. The mean of the RMSE values
    iii. The standard deviation of the RMSE values

Part 3: Decision Tree Regressor Tuning.

a) Print the default hyperparameter settings of a decision tree regressor.
b) Identifying a decision tree that doesn't overfit and generalizes to external data is equivalent to limiting the depth to which it is grown. Two hyperparameters (*max_depth* and *min_samples_leaf*) allow for the depth to

be limited, but they need to be considered independently. Conduct ***two independent*** searches employing *GridSearchCV* with:

i) Estimator= decision tree regressor from Part 2a, scoring= *neg_root_mean_squared_error,* and Parameter grid with *max_depth = 1-19*

ii) Estimator= decision tree regressor from Part 2a, scoring= *neg_root_mean_squared_error,* and Parameter grid with *min_samples_leaf: 1-50*
Hint: The parameter grid can be constructed such that a single fit of *GridSearchCV* can be called to solve the two independent searches (Consider a list of dictionaries…).

c) Identify the optimal hyperparameter settings.
If a single search over two grids was executed, print the best score and corresponding hyperparameter setting. If two separate searches were executed, print the best scores and corresponding hyperparameter setting from each grid search. In a Markdown cell, document the best overall score and hyperparameter setting from the two searches.

d) Using the cv_results dictionary, pair the hyperparameter values and the corresponding RMSE for each hyperparameter setting combination considered by the grid search and print them.
Hint: Look for 'params' and 'mean_test_score' within cv_results_. The 'mean_test_score' is the average of the negated root mean squared error calculated during cross validation of each hyperparameter combination.

Part 4: Random Forest Regressor Tuning.

a) Print the default parameters of this random forest regressor.

b) Using *GridSearchCV*, conduct a ***single*** search with:
Estimator= random forest regressor from Part 2a, scoring= *neg_root_mean_squared_error,* and parameters: *n_estimators*:[100, 200, 400, 500] and *max_features*:[2, 4, 6, 8, 10]

c) Print out the best hyperparameter settings and best score found from this grid search

d) Using the cv_results dictionary, pair the hyperparameter values and the corresponding RMSE for each hyperparameter setting combination considered by the grid search and print them.
Hint: Look for 'params' and 'mean_test_score' within cv_results_. The 'mean_test_score' is the average of the negated root mean squared error calculated during cross validation of each hyperparameter combination.

e) Identify the feature importances from the best estimator, pair the importance values with their corresponding feature, and print the feature, importance pairs.

f) Describe how the importances compare to the correlation matrix implemented in Part 1c.

Part 5: Model Evaluation and Selection

a) Compare the average RMSE values from the following trained challenger models:

i) The four models in Part 2
   1) Linear Regressor with entire training data
   2) Decision Tree Regressor
   3) Random Forest Regressor
   4) Linear Regressor with the single best feature found in Assignment 1

ii) The tuned decision tree model from Part 3 of this model.
Tip: Review the scores from Part 3c to identify the optimal RMSE from the grid search.

iii) The tuned random forest model from Part 4 of this model.
Tip: Review the scores from Part 4c to identify the optimal RMSE from the grid search.

b) Of the models considered in Part 5a, document the model being selected as the champion.

c) With the best performing model identified, make predictions with the ***TEST*** set, and calculate the RMSE from the test set predictions. Document how this RMSE compares to the average RMSE found from the corresponding algorithm from cross validation or grid search that used the training data (RMSE scores from Parts 2, 3, or 4).
Hint: Two search methods were considered in this assignment. If the model selected in Part 5b was found from the average metric using *cross_val_score*, the model needs to be fit to the entire training data. If the best model was found from *GridSearchCV*, the grid search object can make predictions without fitting to the entire training data.