INFO6105 - DSEM

Assignment 2

Author: Aditya Mulik
NUID: 002127694
Email ID: mulik.a@northeastern.edu

Notes:

1. I have used Google Colab for running the jupyter files
2. The datasets are loaded in Google Colab using it's Gdrive API
3. I have linked the dataset or uploaded it in the zip file as per respective question and mentioned in its respective notes

---

Question 1: 1. Try different imputation methods on the Titanic Dataset, and evaluate classifier accuracies for each of these. How would you pick the most optimal imputation method?

I. Different Imputation methods on the Titanic Dataset and the classifier accuracies

No	Imputation Method	Applied On	Accuracy
1	median	Age, Fare	0.56
2	mean	Age, Fare	0.56
3	Mode (Most frequent)	Age, Fare	0.55
4	Random Value (Value not from the continuous values range)	Age, Fare	0.54

II. How to pick the most optimal imputation method?

- By analysing the data and checking the columns with missing values
- Setting their status to missing with another column
- Impute the columns with different methods and compare the prediction accuracies
- Select the imputation method with the lowest overall error value
- Check for the RMSE (root-mean-square-error) to better identify the imputation method

Note: Question_1_DifferentImputationMethods.ipynb file attached
Dataset: titanic_full.csv attached

Question 2: Briefly describe how gradient boosting differs from bagging. Compare the performance of kNN, random forest classifier and gradient boosting on the Titanic dataset.

I. Gradient Boosting vs Bagging

No	Gradient Boosting	Bagging
1	Gradient Boosting is a method of merging different types of predictions	Bagging decreases variance and solves over-fitting issues in a model
2	Connecting predictions of different types can be carried out	Connecting predictions of same types can be carried out
3	New models are affected by the performance of the previous model	Every model is constructed independently
4	Models are weighted by their performance	Every model receives an equal weight

II. Comparison of performance of accuracies of KNN Classifier, Random Forest Classifier and XGBoost (gradient boosting)

No	Algorithm	Accuracy/ Performance
1	K-nearest Neighbours Algorithm Classifier (n_neighbours=3)	0.81
2	Random Forest Classifier (n_estimators = 1000,max_features = 0.25)	0.74
3	XGBoost Classifier	0.83

Note: Question_2_GradientBoosting.ipynb file has been attached
Dataset: Dataset used is Titanic_full.csv as attached in the zip

---

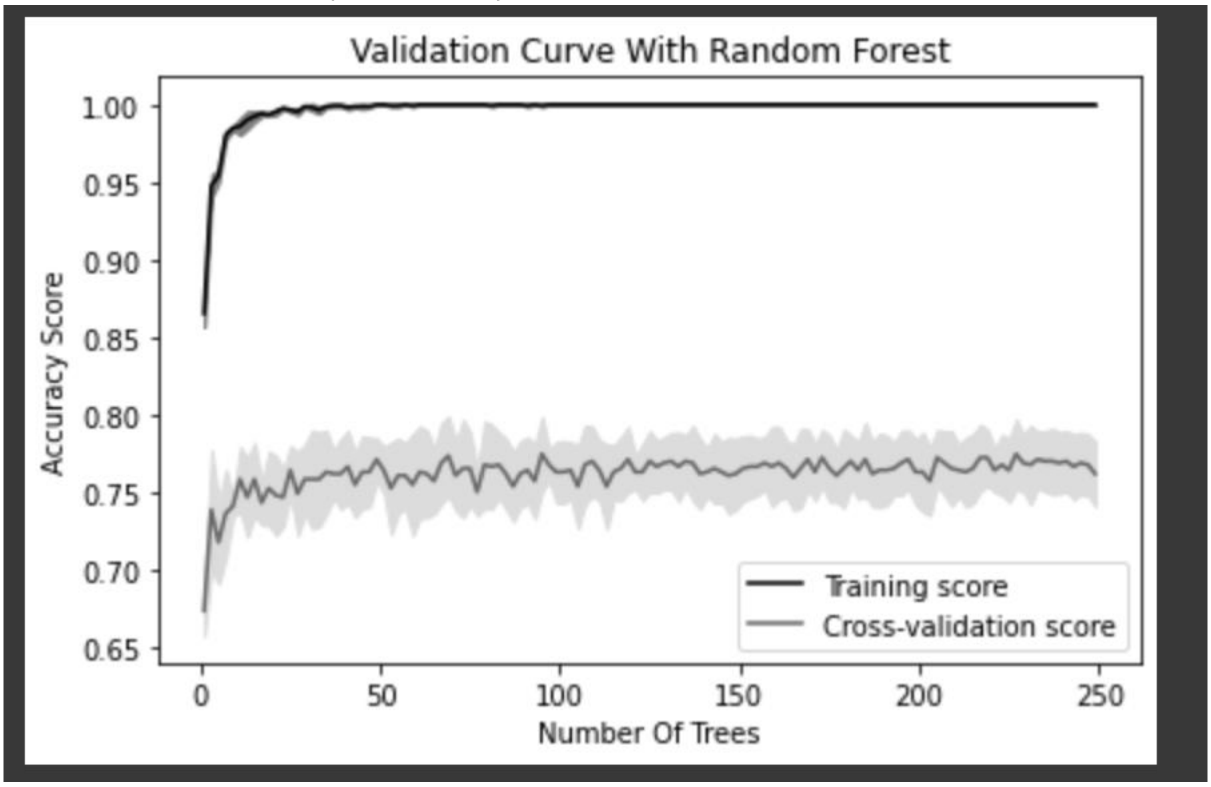
Question 3: Theoretically, increasing the number of decision trees (n_estimators), increases classifier performance and/or generalizability. How would you design and evaluate a computational experiment to test this, on the Titanic dataset? Do you find the same relationship between n_estimators and performance? If not what might be one reason?

I. How would you design and evaluate a computational experiment to test this, on the Titanic dataset?

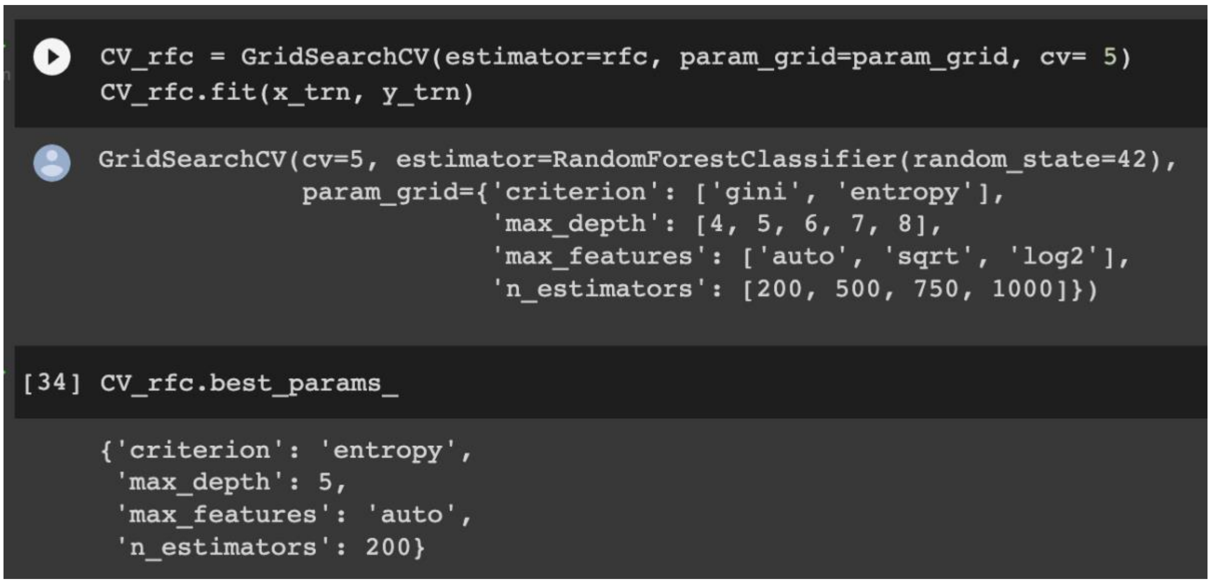
The way to choose the best n_estimators can be done via hyperparameter tuning using manual manipulation of arguments by using various combinations of values to find the best estimate of the n_estimators. One such way would be using a for loop in a range of numbers to try different estimates. But that won't be efficient enough to find the ideal score and it's time consuming.

But the most ideal way would be using several libraries such as below. I have attached the result screenshot after executing the functions to generate the best possible n_estimators

1. Validation_curve (from sklearn)



2. GridSearchCV



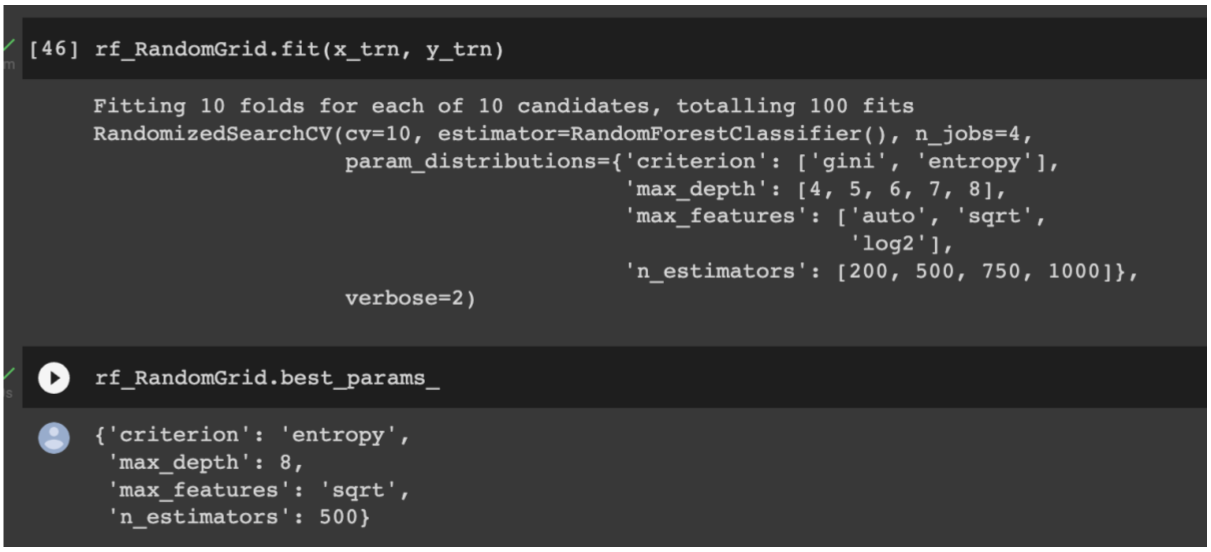
```
CV_rfc = GridSearchCV(estimator=rfc, param_grid=param_grid, cv= 5)
CV_rfc.fit(x_trn, y_trn)
```

```
GridSearchCV(cv=5, estimator=RandomForestClassifier(random_state=42),
             param_grid={'criterion': ['gini', 'entropy'],
                         'max_depth': [4, 5, 6, 7, 8],
                         'max_features': ['auto', 'sqrt', 'log2'],
                         'n_estimators': [200, 500, 750, 1000]})
```

```
[34] CV_rfc.best_params_
```

```
{'criterion': 'entropy',
 'max_depth': 5,
 'max_features': 'auto',
 'n_estimators': 200}
```

3. RandomizedSearchCV



```
[46] rf_RandomGrid.fit(x_trn, y_trn)

     Fitting 10 folds for each of 10 candidates, totalling 100 fits
     RandomizedSearchCV(cv=10, estimator=RandomForestClassifier(), n_jobs=4,
                        param_distributions={'criterion': ['gini', 'entropy'],
                                             'max_depth': [4, 5, 6, 7, 8],
                                             'max_features': ['auto', 'sqrt',
                                                              'log2'],
                                             'n_estimators': [200, 500, 750, 1000]},
                        verbose=2)

rf_RandomGrid.best_params_

{'criterion': 'entropy',
 'max_depth': 8,
 'max_features': 'sqrt',
 'n_estimators': 500}
```

II. Do you find the same relationship between n_estimators and performance? If not what might be one reason?

From the above results, it’s evident that the training score for n_estimators is good, but on executing the same n_estimators (no of trees, even if increased at larger scale) the result (accuracy) is not that great. Increasing the number of trees is not an optimal solution as it results in overfitting on comparing the train vs test accuracy results.

But making use of RandomizedSearchCV is much more powerful as it tries to find the most optimal n_estimators score for the test dataset.

Note: Question_3_Different_n_estimators_titanic_dataset.ipynb file attached
Dataset: titanic_full.csv attached

Question 4: Pick any Kaggle regression dataset. Train, tune and evaluate performance of a Random Forest Regression model. How will you use the feature importance calculations to perform feature selection? Please demonstrate this using the Kaggle regression dataset you picked.

Kaggle Dataset:
https://www.kaggle.com/c/house-prices-advanced-regression-techniques/

Accuracy/ Performance: 0.86

I. How will you use the feature importance calculations to perform feature selection?

- It provides the coefficient values of each feature and it helps find the model's score which best fits for the different features provided
- This will help us choose the best feature to be selected
- Having the best feature selected improves the performance of the model on the new upcoming datasets to get accurate results

Note: Question_4_house_prices_regression.ipynb file attached