COM6012 Assignment 2

Question 1: Using classic supervised learning algorithms

1A:

Using a random 1% of the data for finding the best configuration of parameters

```
Q1_sampled = Q1_rawdata.sample(False, 0.01, 42).cache()
(Q1subset_train, Q1subset_test) = Q1_sampled.randomSplit([0.7, 0.3], 21)

Q1subset_train.write.mode("overwrite").parquet('../Data/Q1subset_training.parquet')
Q1subset_test.write.mode("overwrite").parquet('../Data/Q1subset_test.parquet')
subset_train = spark.read.parquet('../Data/Q1subset_training.parquet')
subset_test = spark.read.parquet('../Data/Q1subset_test.parquet')
```

1B:

An example of a grid search for one of the models (others found in the python file)

```
RF_paramGrid = ParamGridBuilder() \
    .addGrid(RF.maxDepth, [1, 5, 10]) \
    .addGrid(RF.maxBins, [10, 20, 50]) \
    .addGrid(RF.numTrees, [1, 5, 10]) \
    .addGrid(RF.featureSubsetStrategy, ['all','sqrt', 'log2']) \
    .addGrid(RF.subsamplingRate, [0.1, 0.5, 0.9]) \
    .build()

RF_crossval = CrossValidator(estimator=RF_pipeline,
                             estimatorParamMaps=RF_paramGrid,
                             evaluator=MulticlassClassificationEvaluator(),
                             numFolds=5)
```

1C:

Used the same splits of training and test data for each model

```
Q1subset_train.write.mode("overwrite").parquet('../Data/Q1subset_training.parquet')
Q1subset_test.write.mode("overwrite").parquet('../Data/Q1subset_test.parquet')
subset_train = spark.read.parquet('../Data/Q1subset_training.parquet')
subset_test = spark.read.parquet('../Data/Q1subset_test.parquet')
```

2A:

Used the best parameters from the previous step (shown in code using .bestModel)

2B:

Also used the same splits of training and test data (also shown in code)

2C:

Training times:

	5 Cores Training time (s)	10 Cores Training time (s)
Random Forest	**11.57**	**11.54**
Gradient Boosting	**19.22**	**21.54**
Shallow Neural Network	**12.35**	**9.14**

3.

- **Random Forest 3 most relevant features:** mbb, mwbb, mjlv
- **Gradient Boosting 3 most relevant features:** mlv, mwbb, mjlv

4:

Observation 1: The shallow neural network performed considerably worse than the other two models with much lower accuracy and area under the curve. This may be due to the fact that the hyperparameters in neural networks require more testing to be effectively tuned to form a good predictive model.

Observation 2: The random forest and gradient boosting models performed very similarly with almost identical performance metrics of 0.70 for Random Forest and 0.71 for Gradient boosting.

Observation 3: The 3 most relevant features are very similar for the different models. With both 'mwbb' and 'mjlv' present and the only difference being 'mbb' for Random Forest instead of 'mlv' for Gradient Boosting.

Question 2: Senior Data Analyst at Intelligent Insurances Co.

1A:

I calculated the modal value of all of the features containing missing values and found that for Cat7 the modal class was missing values, as a result I decided that it would be best to drop this column entirely. Furthermore, for the other features containing missing values they contained a much lower proportion of missing values so I decided it would be useful to impute these missing values with the modal value as shown in the code.

1B:

Converted the categorical variables using a string indexer followed by a one hot encoder.

```
cat_cols = ['Blind_Make', 'Blind_Model', 'Blind_Submodel', 'Cat1', 'Cat2', 'Cat3', 'Cat4', 'Cat5',\
 'Cat6', 'Cat8', 'Cat9', 'Cat10', 'Cat11', 'Cat12','NVCat', 'OrdCat']
stage_string = [StringIndexer(inputCol=c, outputCol=c+"_index") for c in cat_cols]
stage_ohe = [OneHotEncoder(inputCol=c+"_index", outputCol=c+"ohe") for c in cat_cols]
```

1C:

Balanced the dataset by undersampling the majority class i.e. undersampling samples where the claim amount is equal to 0. This was done by randomly selecting samples of the majority class to balance the training set. As can be seen in the code

2A:

MAE: 11659

MSE: 95

2B:

5 Cores training time: 7.21 seconds

10 Cores training time: 7.89 seconds

3A:

Used a DT as the binary classifier to tell whether the claim is zero or different from zero. See code

3B:

MAE: 214 699

MSE: 189.45

3C:

5 Cores training time: 25.12 seconds

10 Cores training time: 23.56 seconds

4:

Observation 1: The dataset is full of missing values meaning that simply dropping all the missing values would severely diminish the size of the dataset. This is why I decided to use the mode to impute missing values.

Observation 2: The majority of claims were zero with a very small fraction of the dataset containing claims larger than zero. This made it very difficult to construct an effective model without overfitting for the majority class, which is why undersampling was used.

Observation 3: The two models in tandem performed much worse than the simple linear regression model with a considerably higher MAE and MSE. This shows that the tandem model struggled to accurately predict the correct claim amounts when they were greater than zero.