AIAP Machine Learning Problem Report

1. Exploratory data analysis:

From the given dataset, we can conclude a few issues:

- NaN values
- Presence of outliers in ‘models’ and ‘params’ columns

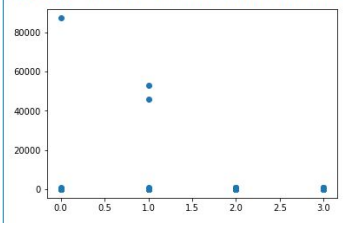
The above is found using the .info() and .describe() methods:

```
In [308]: data.info()
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4331 entries, 0 to 4330
Data columns (total 9 columns):
Unnamed: 0    4331 non-null int64
type          3905 non-null float64
time          4331 non-null float64
dayofweek     4331 non-null float64
models        4331 non-null float64
params        3989 non-null float64
queuelen      4331 non-null float64
trials        4331 non-null float64
duration      4331 non-null float64
dtypes: float64(8), int64(1)
memory usage: 304.6 KB
```

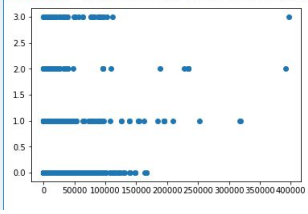
```
In [309]: data.describe()
Out[309]:
       Unnamed: 0         type         time    dayofweek       models  \
count  4331.000000  3905.000000  4331.000000  4331.000000  4331.000000
mean   2165.000000     9.288092     0.572203     3.236666    71.756638
std    1250.396337     3.625067     0.165923     1.572161  1701.084200
min       0.000000     0.000000     0.000694     1.000000     1.000000
25%    1082.500000     7.000000     0.454167     2.000000     5.000000
50%    2165.000000     9.000000     0.584722     3.000000    13.000000
75%    3247.500000    13.000000     0.691667     5.000000    26.000000
max    4330.000000    14.000000     0.999306     7.000000  87293.000000

              params     queuelen       trials     duration
count    3989.000000  4331.000000  4331.000000  4331.000000
mean    10901.674354    53.383514    58.479104     0.728007
std     26059.953210   355.957245    68.839108     0.891271
min        71.000000     0.000000     5.000000     0.000000
25%       952.000000     0.000000    40.000000     0.000000
50%      2995.000000     0.000000    40.000000     0.000000
75%      6940.000000     0.000000    40.000000     1.000000
max    396701.000000  5605.000000   400.000000     3.000000
```

We observe that ‘models’ column has a max of 87 293 whereas the median is only 13. There is this level of discrepancy in the ‘params’ column as well. We can see in a scatter plot against duration where these are more easily identifiable:

```
In [310]: plt.scatter(data.duration, data.models)
Out[310]: <matplotlib.collections.PathCollection at 0x17220de08d0>
```




```
In [311]: plt.scatter(data.params, data.duration)
Out[311]: <matplotlib.collections.PathCollection at 0x17216c7f208>
```



2. Feature engineering

This portion involved the necessity to fill NaN values and to drop the outlier data to improve the model prediction scores. NaN data is present in ‘type’ and ‘params’ columns. For ‘type’ column the NaN values are filled with a forward fill as median results in the ‘type’ with value of 9.0 which is the highest frequency. This could lead to biased data. Whereas for ‘params’ column it is filled with median values. The main decision on this is that the median score reflects a suitable candidate for the ‘params’ value once the outliers are dropped. Outlier row indexes are found through the .idxmax() method. Once these issues are taken care of, we have our new dataset **data_new** from which we can apply our various models and finetune it to arrive at the most accurate one.

3. Modelling & Evaluation

This initial chosen model was a Logistic Regression model as we needed to determine the 'duration' variable which has a few categories. Hence, a logistic regression would work better as we are determining the classification of the 'duration' variable as opposed to a value. This is also seen from the poor prediction score when employing a linear regression model (32-35%). However, after running with a logistic regression model and with a test/train ratio of 0.1, the prediction score is only above average at roughly 60-62%.

To improve the accuracy, we can use a gradient boost algorithm from sklearn library. This does improve the prediction scores as we notice a boost of roughly 10% or more from the prediction scores.

However, the best performance comes from the usage of a decision tree model. This model has consistent prediction scores at 80% and this is a 20% improvement from the initial chosen model of logistic regression. Thus, this should be the model of choice for this problem to determine the 'duration' variable. The confusion matrix arrays for logistic regression and decision tree are seen below:

```
LinearRegression Score: 0.34731161961678525
Logistic Matrix:
[[223   9   0   0]
 [ 96  29   0   0]
 [ 23  14   5   6]
 [  9   7   1  11]]
LogisticRegression Score: 0.6189376443418014
Boosted Score: 0.7299539716978133
Tree Matrix:
[[205  23   3   1]
 [ 21  91  12   1]
 [  4  13  27   4]
 [  0   2   3  23]]
Tree Score: 0.7990762124711316
```

4. Conclusion

In conclusion, although a gradient boosting algorithm may boost the prediction scores of a model, it is in the best interest of the individual to consider the best model for the task at hand. In this case, a decision tree is likely to offer the best results as it results in prediction scores of at least 80%.

Overall, this has been a fruitful foray into the world of data science and machine learning with an interesting case study involving the prediction of a variable. It's always interesting to deal with a real problem to try and get a feel of the thought processes and decisions involved. It is clear that although models make our life easier, bulk of the work is actually dependant on the data preparation aspect. (They should call us tailors instead =)