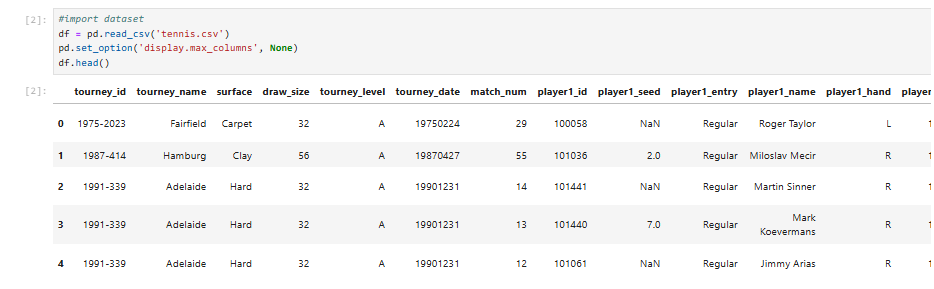
**MS9005/MS9009 Milestone 2**

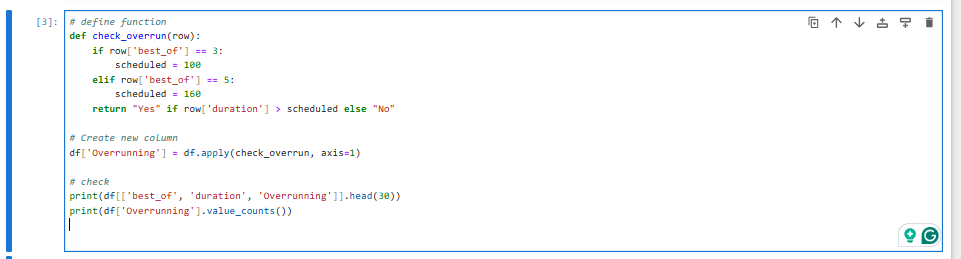
***Dataset Used***

We will use the data where we left off. Hence we will upload the saved CSV File.



***Create Binary Variable***

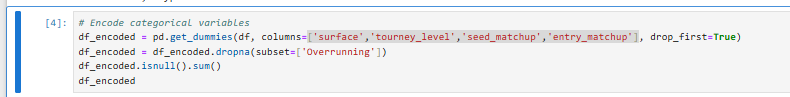
Next we will create a binary variable ‘Overruning’ to indicate if the particular tennis match overran on the scheduled durations (100 Minutes for best of 3, 160 Minutes for best of 5).



***Encoding Qualitative Variables***

We will also encode the following variables for us to use in the respective models:

* 'Surface',
* 'Tourney\_level',
* 'Seed\_matchup'
* 'Entry\_matchup'



***Deciding which Features to Use***

Based on previous models, we will only use variables that were more likely to determine the outcomes of match durations. As such, we will be using the following variables:

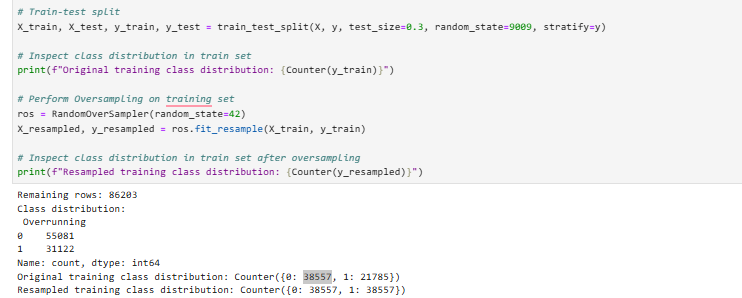
* 'Surface',
* 'Tourney\_level',
* 'Seed\_matchup'
* 'Entry\_matchup'
* "age\_diff",
* "rank\_diff",
* "best\_of"

**Addressing Class Imbalance**

For our dataset, there are a total of 55081 matches that did not overrun and 31122 matches that overran. This shows that there is a class imbalance that may affect the outcome of our models. To address this, we will perform random oversamping of our training dataset.

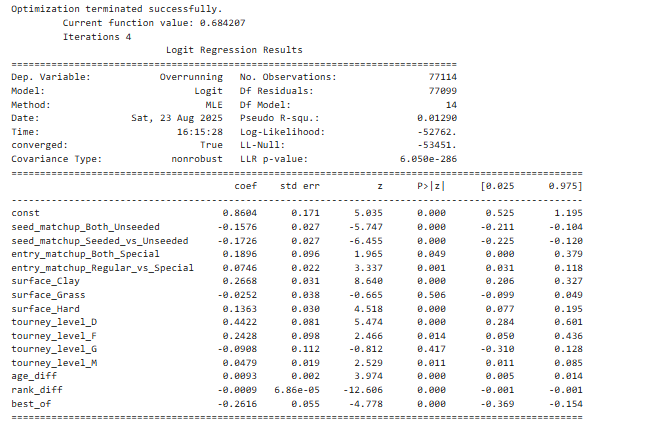
When we do a 70-30 split the on our data, we get 38557 matches that did not overrun, and 21785 matches that overran.

After random oversampling, we have 38557 matches each for both classes.



1. ***First Model: Logistic Regression***

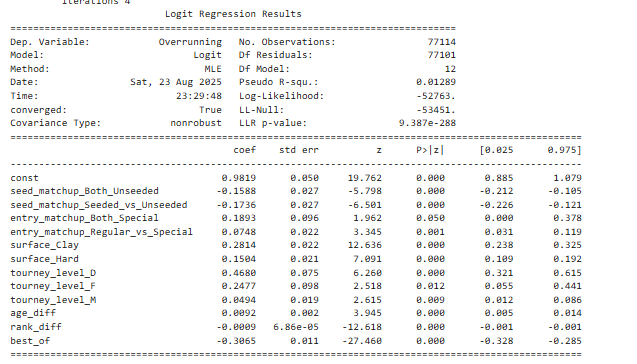
For our logistic regression model, we will fit the training dataset prepared above. After which, we can observe the model summary as shown below



***Fine tuning the model***

After inspecting the model summary and the p-values of each variable, we can remove features of the model that are statistically insignificant. This will

Hence, ‘surface\_grass’ and ‘tourney\_level\_g’ can be removed from the model. After they have been removed we will inspect the model summary shown below again.



Now we see that all our variables are statistically significant and are useful to our model.

***Evaluating the model***

We will use this model to predict if the durations of matches overran in our test dataset. The confusion matrix and Scoring metrics are shown as such:

Confusion Matrix:

[[ 6349 10175]

[ 2540 6797]]

Accuracy: 0.508333011097792

Precision: 0.4004831487155315

Recall: 0.7279640141373032

F1 Score: 0.5167053099699722

The logistic regression model achieved an overall accuracy of 50.8%, The model demonstrated a high recall (72.8%), indicating that it correctly identified most of the matches that were overrunning. However, it also had very low precision (40.0%), meaning that many non-overrunning matches were incorrectly classified as overrunning. The model is more effective at detecting overruns but generates a significant number of false positives, which reduces its reliability.

1. ***Second Model: Decision Tree***

We will also fit a decision tree model on our training data. We will use Gini Index, without specifying the parameters for the initial model. The confusion matrix and Scoring metrics are shown as such:

Confusion Matrix:

[[10724 5800]

[ 5811 3526]]

Accuracy: 0.5510227756080585

Precision: 0.37808277932661377

Recall: 0.37763735675270427

F1 Score: 0.37785993677329477

***Fine Tuning the Model***

As we did not add any parameters to the model, the model might be overfitted and will not work well on testing data.

To improve the decision tree model, a Grid Search with 5-fold cross-validation was conducted using the gini criterion. The search tested different values for maximum depth, minimum samples required to split, minimum samples per leaf, and pruning strength. The model was evaluated based on accuracy, and the parameter combination with the best performance was selected.

We find the best parameter combination from the output below.

Best Params: {'ccp\_alpha': 0.0, 'max\_depth': 10, 'min\_samples\_leaf': 1, 'min\_samples\_split': 2}

**Fitting the Model with Optimum Parameters**

We will now fit the model with the optimum parameters. The confusion matrix and Scoring metrics are shown as such:

Confusion Matrix:

[[6907 9617]

[3061 6276]]

Accuracy: 0.5097637369011252

Precision: 0.39489083244195555

Recall: 0.6721645068008997

F1 Score: 0.49750297265160526

After parameter tuning, the model’s accuracy decreased slightly (50.9% vs 55.1%), but its recall improved substantially (67.2% vs 37.8%). This means the tuned model is much better at identifying overruns (sensitivity) but less accurate overall. The default model is more balanced, while the tuned model prioritizes catching as many overruns as possible, However, it is more likely to predict more false alarms, in this case, predicting overruns when the match will not overrun.

1. ***Third Model: KNN***

We will now use the same training dataset to train the KNN Model. We will first assign k=5 and examine the initial model. The confusion matrix and Scoring metrics are shown as such:

Confusion Matrix:

[[9338 7186]

[4964 4373]]

Accuracy: 0.5301805807973397

Precision: 0.3783199238688468

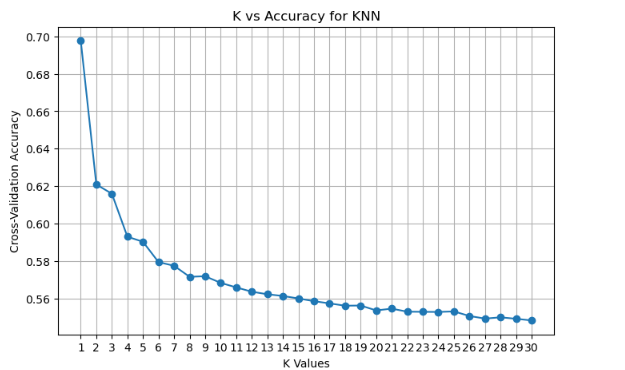
Recall: 0.46835171896754846

F1 Score: 0.4185490045941807

***Fine Tuning the model***

To find the best value of K for the KNN model, we can test values from 1 to 30 using 5-fold cross-validation on the oversampled training data. For each K, the average accuracy will be calculated and plotted.

The graph shows how accuracy changes with different K values. Small K values may overfit the data, while very large K values may underfit.

******

To select the most optimum k-value, we will select the value k at the ‘elbow’ on the graph. Hence we will choose k=6.

We will run the model again this time using k=6 instead of 5.

The confusion matrix and Scoring metrics are shown as such:

Confusion Matrix:

[[11180 5344]

[ 6008 3329]]

Accuracy: 0.5610378562313909

Precision: 0.38383488988815867

Recall: 0.35653850273106996

F1 Score: 0.3696835091615769

While we have managed to achieve the highest accuracy score of 56%, this model struggles with predicting the positive class. Its precision of **38.4%** means that when it predicts a positive outcome, it is wrong more often than it is right.

The recall is also very low at **35.7%**, indicating that the model misses a huge number of actual positive cases. The F1 score of **37.0%** confirms the poor balance between precision and recall. We can conclude that despite its higher accuracy score (compared to other models) the model is generally unreliable.

***Comparing all three models***

We will now compare the scoring metrics of all models.

| Logistic Regression | Decision Tree | KNN |
| --- | --- | --- |
| Accuracy: 0.508333011097792  Precision: 0.4004831487155315  Recall: 0.7279640141373032  F1 Score: 0.5167053099699722 | Accuracy: 0.5097637369011252  Precision: 0.39489083244195555  Recall: 0.6721645068008997  F1 Score: 0.49750297265160526 | Accuracy: 0.5610378562313909  Precision: 0.38383488988815867  Recall: 0.35653850273106996  F1 Score: 0.3696835091615769 |

### ***Ranking of Models***

1. Logistic Regression

This model is the best among all models because of its high F1 score and recall. This indicates a strong balance between identifying positive cases and avoiding false alarms. Hence this is the most reliable model.

1. Decision Tree

While not as strong as Logistic Regression, the Decision Tree is a decent option with a good balance between precision and recall.

1. K-Nearest Neighbors (KNN)

Despite having the highest accuracy (56.1%), KNN is the least reliable model. Its low F1 score shows a severe imbalance, showing it either misses too many positive cases (low recall) or makes too many false positive predictions (low precision).

***Conclusion***

All three models show generally unreliable predictive power, with Accuracy values all around 50–56%. The low precision across models further highlights the risk of generating excessive false positives.

In conclusion, while the models capture some signal in the data, their performance is not reliable enough for confident decision-making. Further work such as more data collection or feature engineering will be needed to significantly improve the prediction power of these models.