Optimizing the *Wheel of Fortune* Bonus Round

Math 37700 Final Capstone

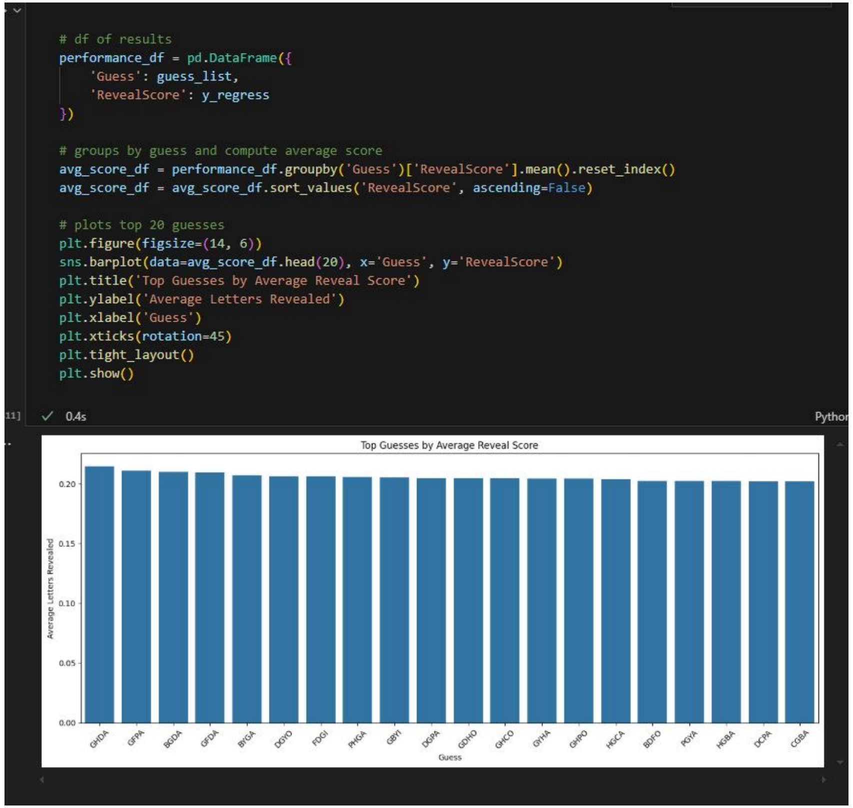
Axyl Fredrick

Problem Overview

- Analysis of a set of *Wheel of Fortune* Puzzles to find the optimal combination of letters in the Bonus Round.
- In the Bonus Round, one picks 3 consonants and 1 vowel after the puzzle has R, S, T, L, N, E in place.
- We are using string a dataset I scraped from a Wheel fan website and put in a CSV file.
- Data consists of Puzzle, Category, and Airdate of Episode.
- Most relevant is the puzzle itself, and it was what all our models were built on.
- In addition to a best case scenario of monetary gain, these findings can be used by the showrunners to measure difficulty of these puzzles.

Statistical Progress

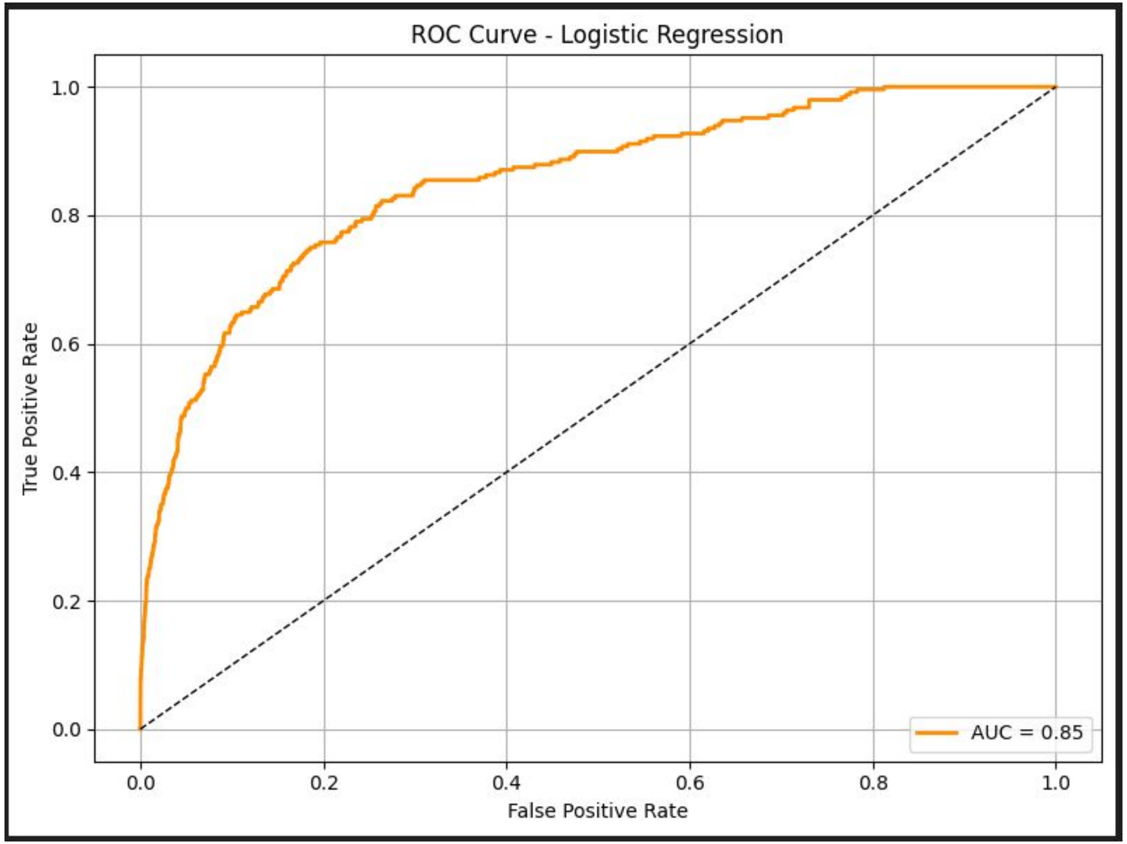
- I have used SciKit learn and a regression model to maximize the percentage of each puzzle that is solved.
- One change made to the project is that I am only using one dataset, which is the Wheel dataset, mainly due to feature constraints and for a better focus on the problem at hand.
- A new model was made with logistic and linear regression combined for an ensemble model.
- On the right is our results with the linear model alone.



```
# df of results
performance_df = pd.DataFrame({
    'Guess': guess_list,
    'RevealScore': y_regress
})

# groups by guess and compute average score
avg_score_df = performance_df.groupby('Guess')['RevealScore'].mean().reset_index()
avg_score_df = avg_score_df.sort_values('RevealScore', ascending=False)

# plots top 20 guesses
plt.figure(figsize=(14, 6))
sns.barplot(data=avg_score_df.head(20), x='Guess', y='RevealScore')
plt.title('Top Guesses by Average Reveal Score')
plt.ylabel('Average Letters Revealed')
plt.xlabel('Guess')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```

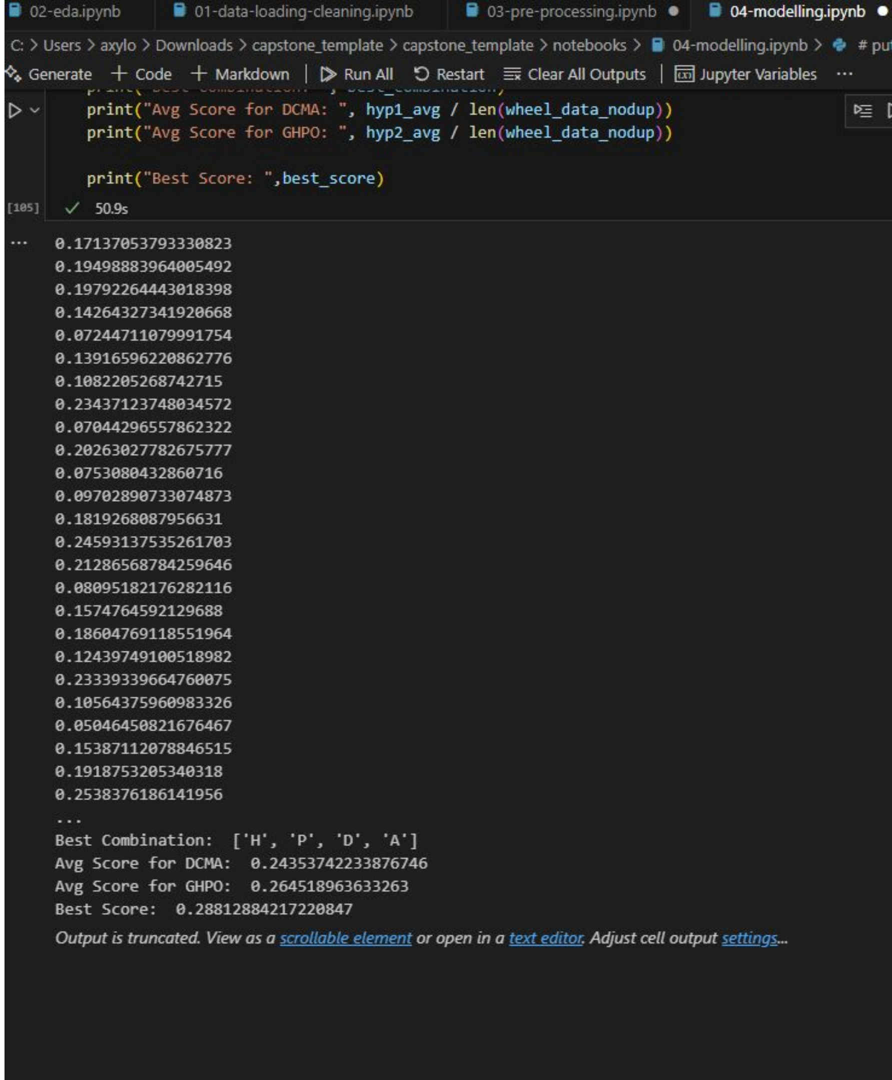
ROC Curve Logistic

- Here is how the ROC curve holds for logistic regression!
- The findings lean away from the line to the upper left, so they are good but could be improved.



Final Calculation

- H, P, D, A is the guess that the model stumbled upon.
- The code could be optimized for better performance, but in terms of functionality it is solid.
- Linear + logistic model. Logistic model fit well.
- The linear and logistic were added by weights.
- Regression with linear + Probability of good guess with logistic


02-eda.ipynb
01-data-loading-cleaning.ipynb
03-pre-processing.ipynb
04-modelling.ipynb
C: > Users > axylo > Downloads > capstone_template > capstone_template > notebooks > 04-modelling.ipynb > # pu
Generate
Code
Markdown
Run All
Restart
Clear All Outputs
Jupyter Variables
[105]
50.9s

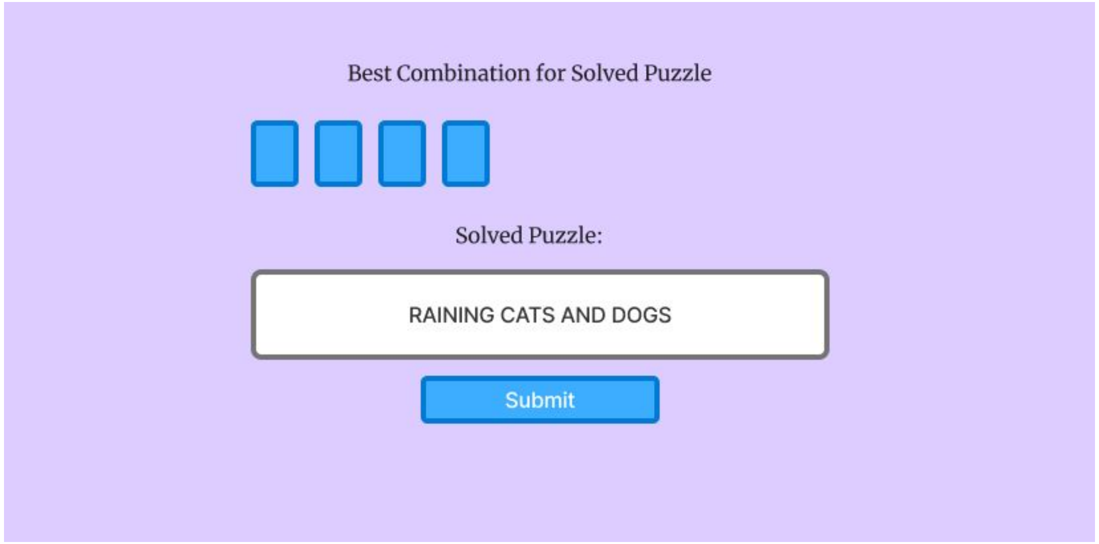
```python
print("Avg Score for DCMA: ", hyp1_avg / len(wheel_data_nodup))
print("Avg Score for GHPO: ", hyp2_avg / len(wheel_data_nodup))

print("Best Score: ",best_score)
```

```
0.17137053793330823
0.19498883964005492
0.19792264443018398
0.14264327341920668
0.07244711079991754
0.13916596220862776
0.1082205268742715
0.23437123748034572
0.07044296557862322
0.20263027782675777
0.0753080432860716
0.09702890733074873
0.1819268087956631
0.24593137535261703
0.21286568784259646
0.08095182176282116
0.1574764592129688
0.18604769118551964
0.12439749100518982
0.23339339664760075
0.10564375960983326
0.05046450821676467
0.15387112078846515
0.1918753205340318
0.2538376186141956
...
Best Combination:  ['H', 'P', 'D', 'A']
Avg Score for DCMA:  0.24353742233876746
Avg Score for GHPO:  0.264518963633263
Best Score:  0.28812884217220847
```

Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings...

The Future:

- We can use the data we collected to look for the best combination of letters for individual puzzles, and vice versa.
- In addition, we can continue refining the models we have as well, possibly implementing a neural net and testing its functionality.


Best Combination for Solved Puzzle
Solved Puzzle:
RAINING CATS AND DOGS
Submit