Formula 1 Neural Network

Dexter Corley

Argyros School of Business & Economics

&

Fowler School of Engineering

CPSC 393 - 01

Dr. Nicholas LaHaye

May 17, 2023

Abstract

The final project for this class pulls together all the models and techniques the class has learned throughout the semester. The project is loosely outlined, and students are instructed to choose a dataset and come up with a problem that can be solved through the implementation of a machine learning algorithm. The project I've selected is to accomplish the task of using a neural network to predict the podium placement each driver will get at each race for the upcoming Formula 1 season.

Introduction

Formula 1 is one of the few sports that relies on data to fuel their success and base important decisions on. As a result, one F1 car will collect upwards of a terabyte of data per race weekend. This leads to a huge pool of data that teams use for future car and strategy improvements. Additionally, Formula 1 collects timing data for each of the sessions in the weekend. A typical weekend consists of 3 practice rounds, a qualifying round, and the actual race. The data collected during each session includes a plethora of fields but most importantly is the grid position, average lap time, and standings. This project's objective is to train a model that understands what the important factors are that contribute to a race victory. Both models are using training data from the years 1990 to 2022. This should lead to a lot of historical information the model can use to base predictions on.

The goal of this project isn't to use car data for the training but to use historical race data along with the driver progression through the season and per track performance. This project took two approaches, a regression based neural network and a multiclass classification neural network. Each having similar inputs but different approaches at the podium predictions. The current approaches are very similar to this and are often used for sports book odds and season projections. Aside from victory predictions, one of the current ML tasks within the sport revolves around strategy. The race strategy is hugely important in the result on the podium and mainly involves when to take a pit stop and what tires to use at what times. Monte Carlo simulations are currently the top approach pit mangers use. Some of the recent research has pointed towards incorporating reinforcement learning to leverage deep network architectures that are able to incorporate more data going towards a smarter strategy. This is important because a race strategy can make or break each teams result and is currently a race to getting these models to come up with better strategies for each team.

Datasets

The data is one of the most important aspects of achieving a successful output. For this project, the data was pulled from the Ergast API which houses a database filled with information on the sport since the 1970s. A big portion of the time spent working on this project was spent engineering the dataset. It was originally comprised of different subsets that contained relevant and irrelevant data that then needed to be joined together and categorical variables dummified. To keep it simple and relevant, the model was trained only on per race data that contained the results and standings for each driver, each race, between 1990 and 2022.

Here are the dataset fields:

- **Year**
- **Round**
- **Altitude**
- **DriverID**
- **Driver_Points**
- **Driver_Standings_Position**
- **Driver_Wins**
- **Grid**
- **Podium**
- **Minutes**
- **Dummy variables for the circuit**

Exploratory Data Analysis

The exploratory data analysis portion of this project revealed many insights, correlations, and outliers. Every successful data science project requires extensive data engineering and then a hands-on EDA process. The EDA process involves combing through the data with visualizations and comparisons looking for insight. The data exploration process taken in this project was no different. Once the data was cleaned and put together to form *Figure 1,* the next step in the process was to look at the distributions and address outliers. *Figures 2-9* show each predictor's distribution. I chose to use boxplots for this. They are easy to read and identify outliers with. The current racing lineup consists of 20 drivers and with that only 20 grid and podium positions. Due to this, it's important that all the training data also only contains 20 drivers, grid positions, and podium spots. Other than that, the data was good to go and could continue. Next, the data was sent through a correlation matrix as seen in *Figure 10* which was used to determine the statistical significance variables had with the dependent variable and with each other. Unfortunately, none of the variables were above a 0.4 significance level with Podium placement. The one holding the 0.4 significance level is the grid position which is in my opinion is the most important factor that determines a podium spot. Additionally, *Figure 11* shows a visualization depicting each race from the specified year comparing the relationship between starting grid position and resulting podium placement for the top 5 drivers. After learning grid and podium had a relationship, this visualization showed it off well. Additionally, *Figure 12 and 13* show histogram boxplots between grid and podium and grid and driver standings position. I really like the way these turned out as the results make sense and it's cool to see the distribution for each position. Showing that a driver will be closer to Podium #1 and #1 in the standings the higher they place on the grid.

Models

Two different models were used to tackle the problem at hand. The first was a standard linear regression neural network and the second was a multiclass classification neural network. As the purpose of this model is to be used over and over for evaluating each race throughout the season, guessing correctly is important. K-fold cross validation was incorporated into both models to help with this (*Figure 16*). The first regression based neural network was trained in 4 folds over 500 epochs. As seen in *Figure 15,* the model had two hidden layers and used a MSE loss function. *Figures 17 & 18* show the validation MAE over each epoch. The course of these epochs led to volatile results and the overall model didn't perform great. The ending results came out to have a MAE score of 4.59 and a loss of 28.1. On average, the model predicted the podium 5 places off the actual results.

Additionally, the other model created to take on this task was a multiclass classification network. In many ways this was like the other except different in some data preparation and the model build. *Figure 19* shows the model with 2 hidden layers using categorical crossentropy as the loss and accuracy as the metric. Over the course of 100 epochs, the model had an extremely low loss at 9.0756e-7 and an accuracy of 0.15. This means that it's getting good at everything leading up to the actual predictions. As the accuracy is at 15%, the model isn't the best for correctly guessing a podium position.

These results aren't optimal but also could be a lot worse. One of the big takeaways I've had from this project is that race predictions are very difficult given the countless externalities of the sport.

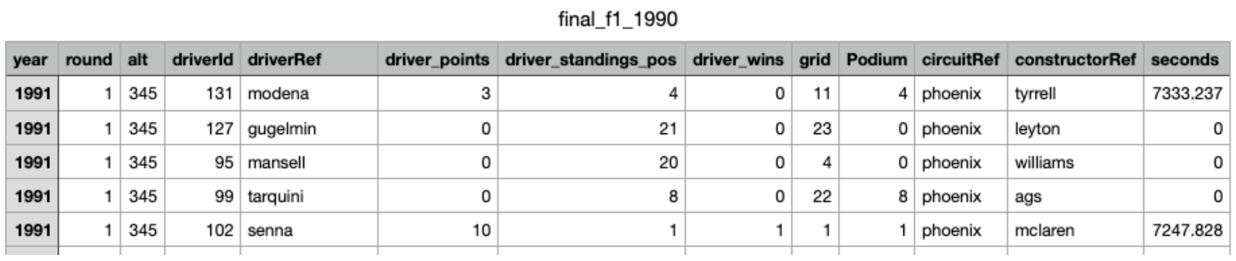
Conclusion

To conclude, the model offered subpar results that I wouldn't recommend for basing predictions on. The results had low accuracy scores and higher than expected MAE scores. As discussed in the previous section, racing is a very difficult sport to predict, and these models didn't do a horrible job. I think given the heatmap and its low correlation values to the podium spots, these models did an ok job. Some considerations in the future would be to build the model deeper. The other thing I'd like to add is that a per track model could do better at predicting the results due to there being less external factors in consideration.

Having a do-everything model for each track inhibits the performance levels as there are a lot of other factors at play not expressed in the data.

Originally it was my goal to use per race, per lap data for every driver. On top of this building out dummy variables for the circuit, constructor, and status (what happened to someone if they DNF'd). I did all the necessary data engineering to make this happen but was left with a dataset containing 210 columns and 530,000 rows. It quickly became apparent that as this was a lot of data the model could use for training, most of it was useless. To conclude, I decided to go with data between 1990 and 2022 and only using per race/driver data which made the model more robust and better trained.

Appendix



final_f1_1990

year	round	alt	driverId	driverRef	driver_points	driver_standings_pos	driver_wins	grid	Podium	circuitRef	constructorRef	seconds
1991	1	345	131	modena	3	4	0	11	4	phoenix	tyrrell	7333.237
1991	1	345	127	gugelmin	0	21	0	23	0	phoenix	leyton	0
1991	1	345	95	mansell	0	20	0	4	0	phoenix	williams	0
1991	1	345	99	tarquini	0	8	0	22	8	phoenix	ags	0
1991	1	345	102	senna	10	1	1	1	1	phoenix	mclaren	7247.828

Figure 1

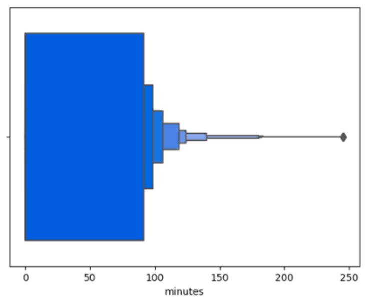


Figure 2

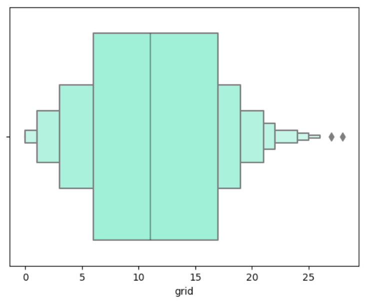


Figure 3

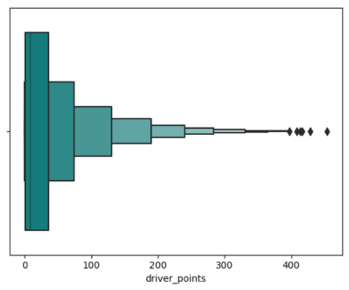


Figure 4

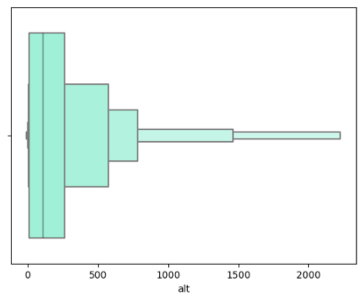


Figure 5

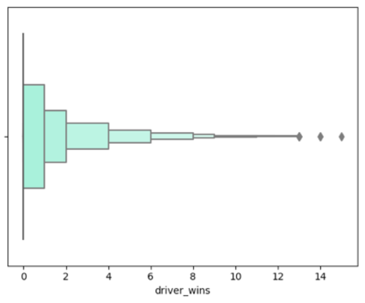


Figure 6

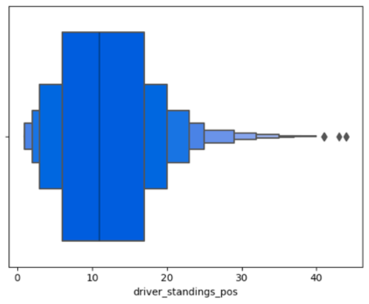


Figure 7

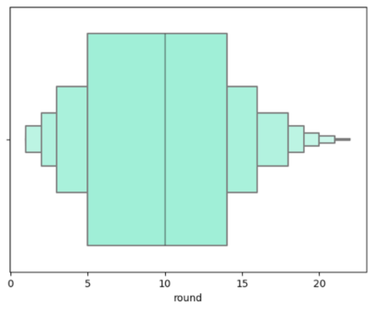


Figure 8

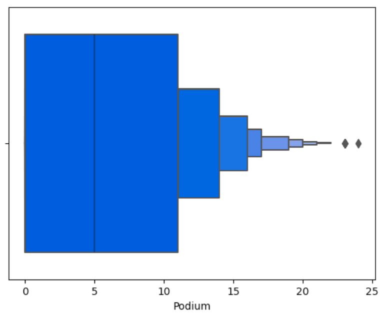


Figure 9

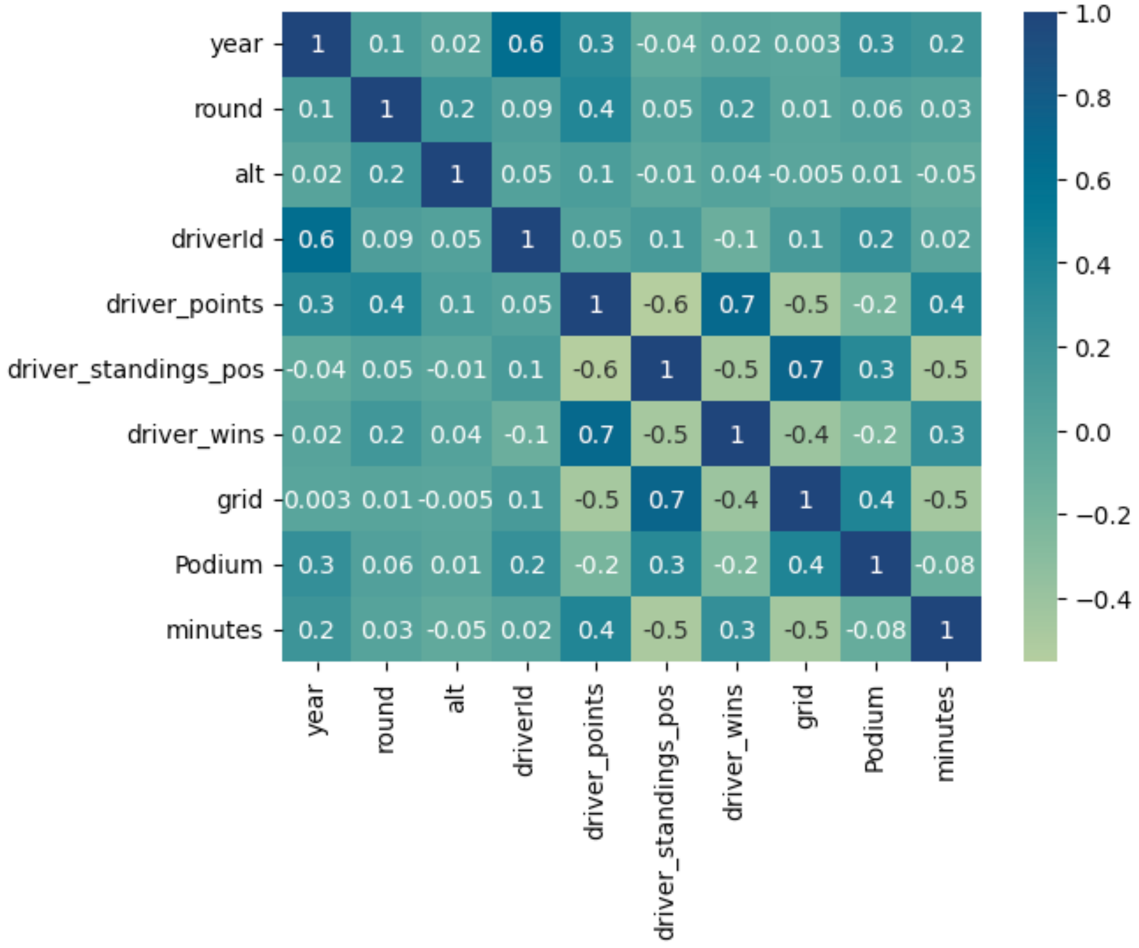

year 1 0.1 0.02 0.6 0.3 -0.04 0.02 0.003 0.3 0.2
round 0.1 1 0.2 0.09 0.4 0.05 0.2 0.01 0.06 0.03
alt 0.02 0.2 1 0.05 0.1 -0.01 0.04 -0.005 0.01 -0.05
driverId 0.6 0.09 0.05 1 0.05 0.1 -0.1 0.1 0.2 0.02
driver_points 0.3 0.4 0.1 0.05 1 -0.6 0.7 -0.5 -0.2 0.4
driver_standings_pos -0.04 0.05 -0.01 0.1 -0.6 1 -0.5 0.7 0.3 -0.5
driver_wins 0.02 0.2 0.04 -0.1 0.7 -0.5 1 -0.4 -0.2 0.3
grid 0.003 0.01 -0.005 0.1 -0.5 0.7 -0.4 1 0.4 -0.5
Podium 0.3 0.06 0.01 0.2 -0.2 0.3 -0.2 0.4 1 -0.08
minutes 0.2 0.03 -0.05 0.02 0.4 -0.5 0.3 -0.5 -0.08 1
year
round
alt
driverId
driver_points
driver_standings_pos
driver_wins
grid
Podium
minutes
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4

Figure 10

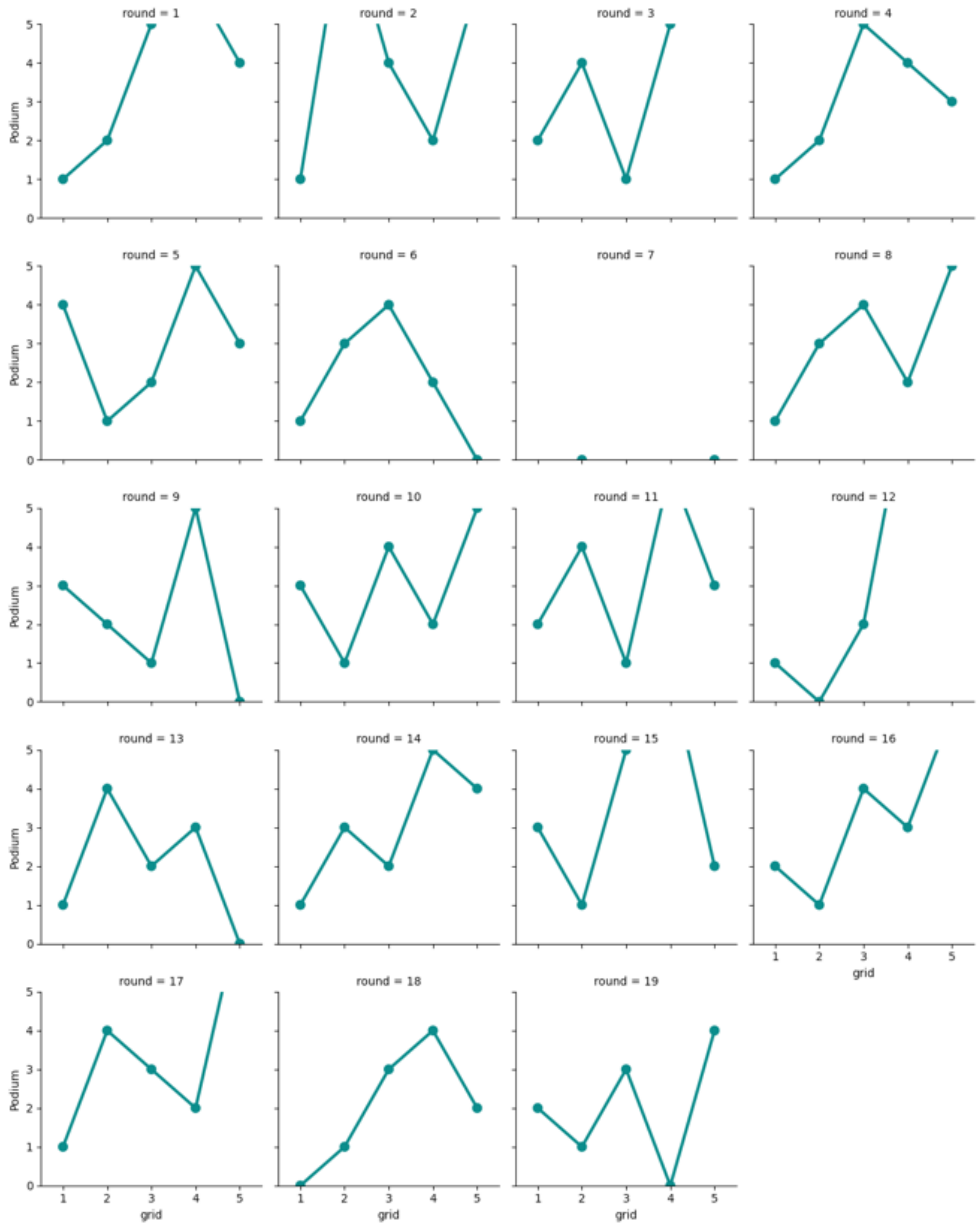

round = 1
round = 2
round = 3
round = 4
round = 5
round = 6
round = 7
round = 8
round = 9
round = 10
round = 11
round = 12
round = 13
round = 14
round = 15
round = 16
round = 17
round = 18
round = 19
Podium
grid

Figure 11

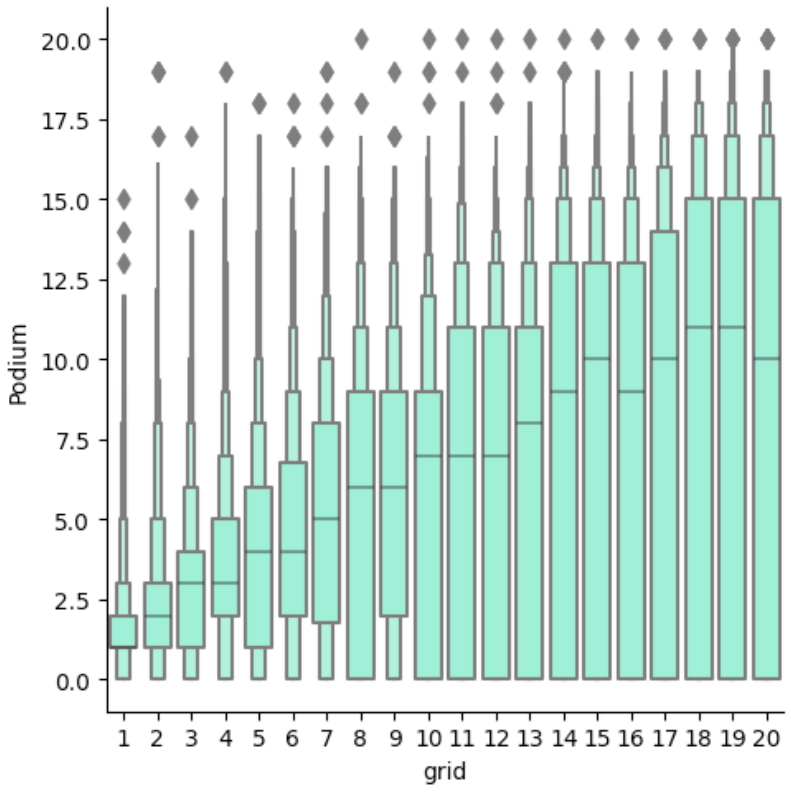


Figure 12

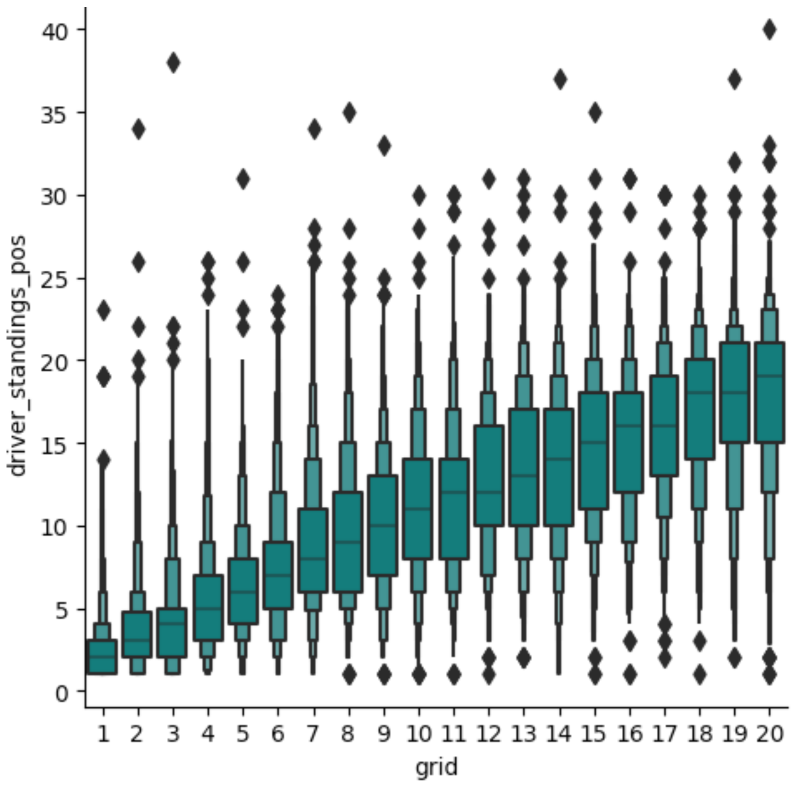
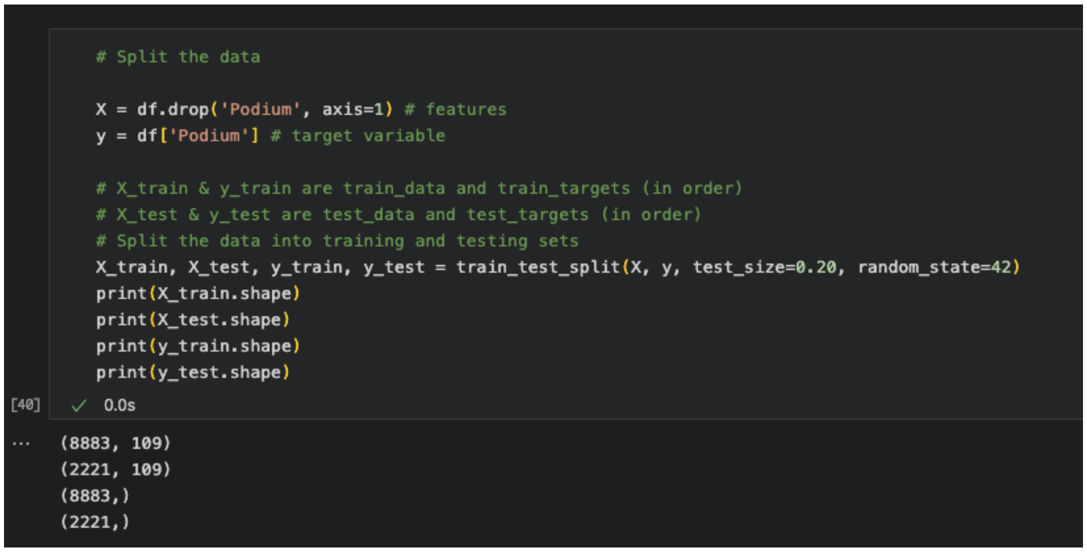


Figure 13

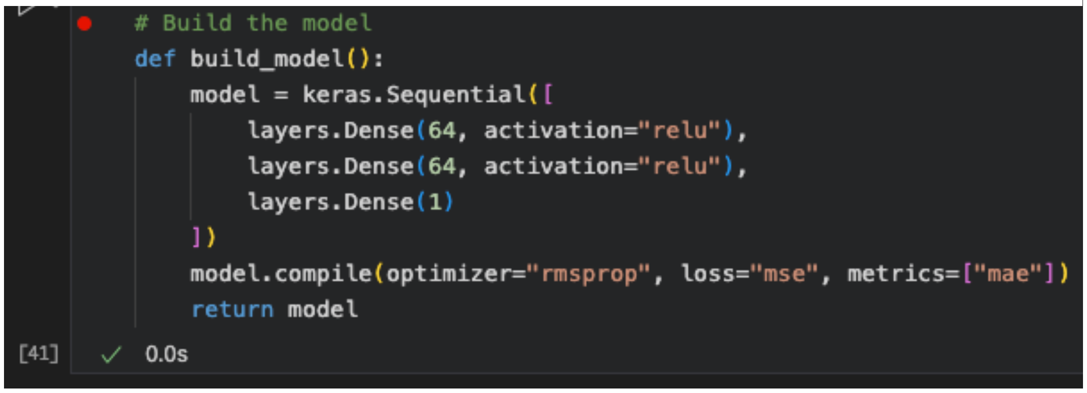


```
# Split the data

X = df.drop('Podium', axis=1) # features
y = df['Podium'] # target variable

# X_train & y_train are train_data and train_targets (in order)
# X_test & y_test are test_data and test_targets (in order)
# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.20, random_state=42)
print(X_train.shape)
print(X_test.shape)
print(y_train.shape)
print(y_test.shape)
```

[40] ✓ 0.0s

```
(8883, 109)
(2221, 109)
(8883,)
(2221,)
```

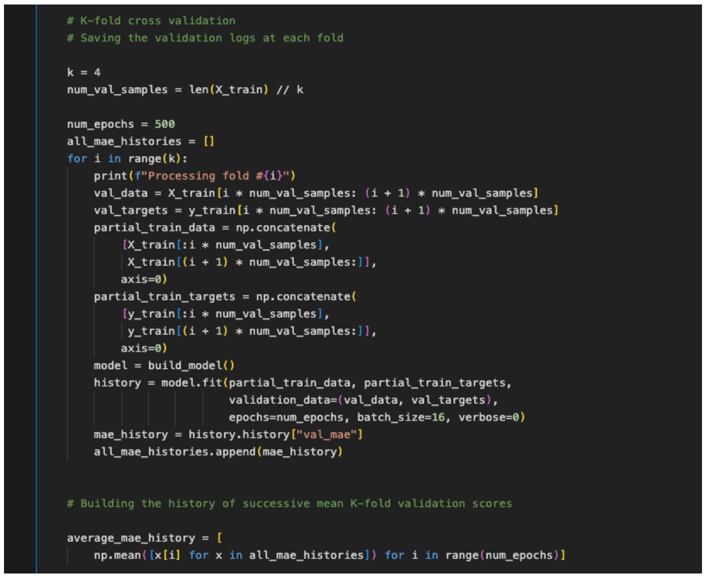
Figure 14



```
# Build the model
def build_model():
    model = keras.Sequential([
        layers.Dense(64, activation="relu"),
        layers.Dense(64, activation="relu"),
        layers.Dense(1)
    ])
    model.compile(optimizer="rmsprop", loss="mse", metrics=["mae"])
    return model
```

[41] ✓ 0.0s

Figure 15



```
# K-fold cross validation
# Saving the validation logs at each fold

k = 4
num_val_samples = len(X_train) // k

num_epochs = 500
all_mae_histories = []
for i in range(k):
    print(f"Processing fold #{i}")
    val_data = X_train[i * num_val_samples: (i + 1) * num_val_samples]
    val_targets = y_train[i * num_val_samples: (i + 1) * num_val_samples]
    partial_train_data = np.concatenate(
        [X_train[:i * num_val_samples],
         X_train[(i + 1) * num_val_samples:]],
        axis=0)
    partial_train_targets = np.concatenate(
        [y_train[:i * num_val_samples],
         y_train[(i + 1) * num_val_samples:]],
        axis=0)
    model = build_model()
    history = model.fit(partial_train_data, partial_train_targets,
                        validation_data=(val_data, val_targets),
                        epochs=num_epochs, batch_size=16, verbose=0)
    mae_history = history.history["val_mae"]
    all_mae_histories.append(mae_history)


# Building the history of successive mean K-fold validation scores

average_mae_history = [
    np.mean([x[i] for x in all_mae_histories]) for i in range(num_epochs)]
```

Figure 16

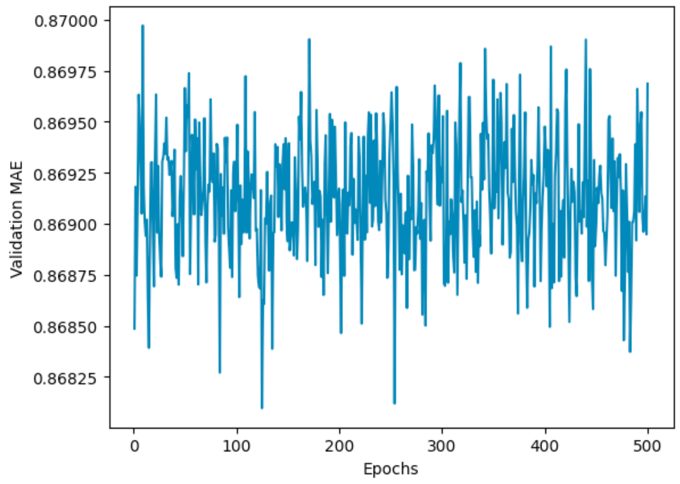


Figure 17

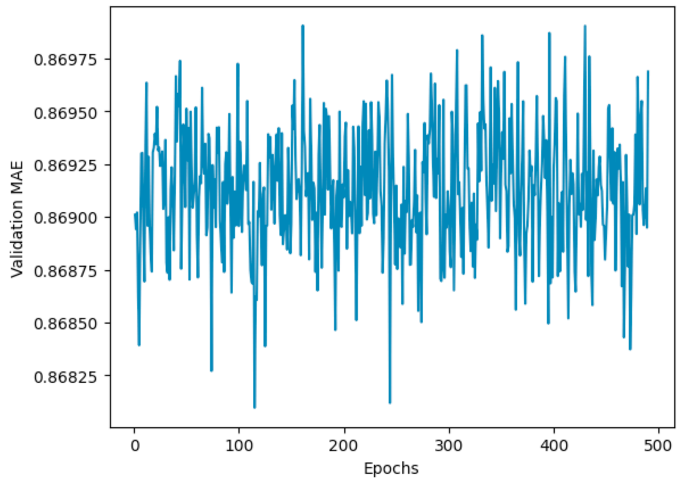
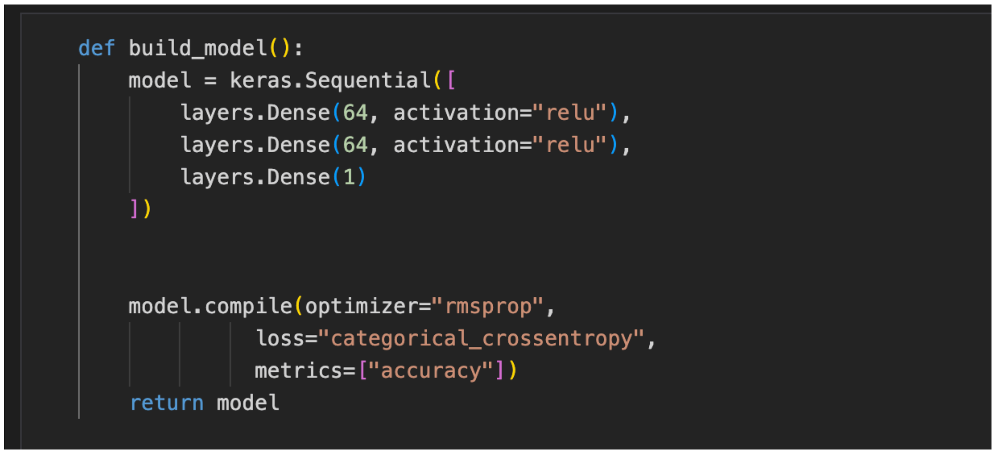
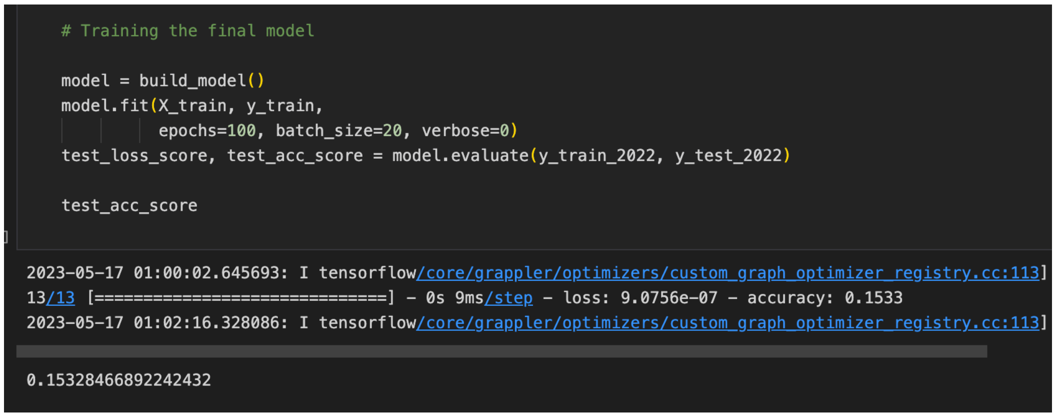


Figure 18



```python
def build_model():
    model = keras.Sequential([
        layers.Dense(64, activation="relu"),
        layers.Dense(64, activation="relu"),
        layers.Dense(1)
    ])


    model.compile(optimizer="rmsprop",
                  loss="categorical_crossentropy",
                  metrics=["accuracy"])
    return model
```

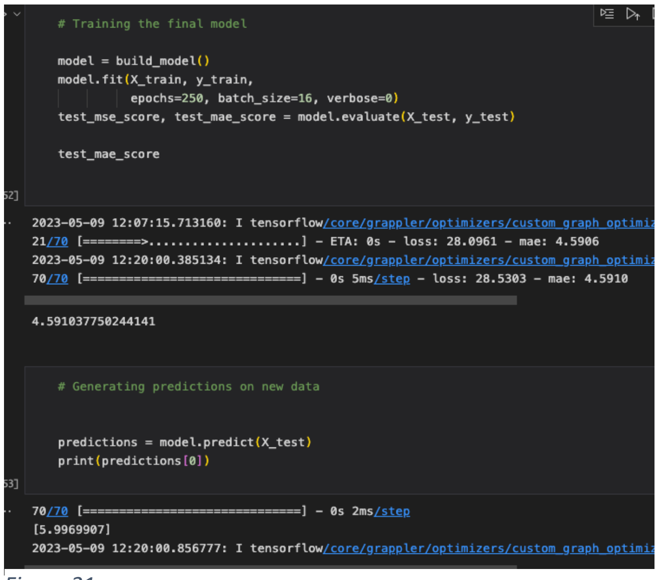
Figure 19



```python
# Training the final model

model = build_model()
model.fit(X_train, y_train,
          epochs=100, batch_size=20, verbose=0)
test_loss_score, test_acc_score = model.evaluate(y_train_2022, y_test_2022)

test_acc_score
```

```
2023-05-17 01:00:02.645693: I tensorflow/core/grappler/optimizers/custom_graph_optimizer_registry.cc:113]
13/13 [==============================] - 0s 9ms/step - loss: 9.0756e-07 - accuracy: 0.1533
2023-05-17 01:02:16.328086: I tensorflow/core/grappler/optimizers/custom_graph_optimizer_registry.cc:113]

0.15328466892242432
```

Figure 20



```python
# Training the final model

model = build_model()
model.fit(X_train, y_train,
          epochs=250, batch_size=16, verbose=0)
test_mse_score, test_mae_score = model.evaluate(X_test, y_test)

test_mae_score
```

```
2023-05-09 12:07:15.713160: I tensorflow/core/grappler/optimizers/custom_graph_optimiz
21/70 [========>.....................] - ETA: 0s - loss: 28.0961 - mae: 4.5906
2023-05-09 12:20:00.385134: I tensorflow/core/grappler/optimizers/custom_graph_optimiz
70/70 [==============================] - 0s 5ms/step - loss: 28.5303 - mae: 4.5910

4.591037750244141
```

```python
# Generating predictions on new data


predictions = model.predict(X_test)
print(predictions[0])
```

```
70/70 [==============================] - 0s 2ms/step
[5.9969907]
2023-05-09 12:20:00.856777: I tensorflow/core/grappler/optimizers/custom_graph_optimiz
```

Figure 21

Citations

Dataset: Via DataIsPlural: https://docs.google.com/spreadsheets/d/1wZhPLMCHKJvwOkP4juclhjFgqIY8fQFMemwKL2c64vk/edit#gid=0

Nigro, Veronica. (2021). Formula 1 Race Predictor. Towards Data Science. Retrieved from https://towardsdatascience.com/formula-1-race-predictor-5d4bfae887da