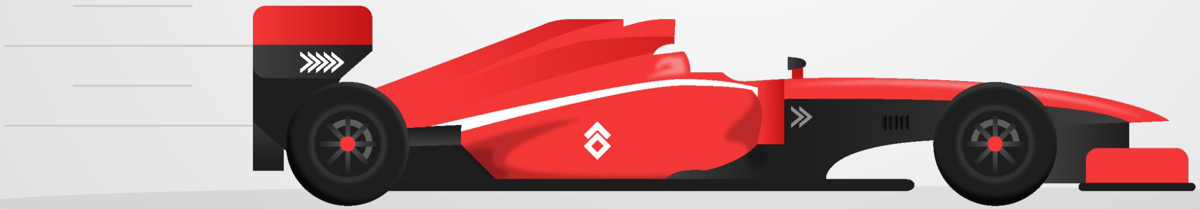
Formula 1 Machine Learning

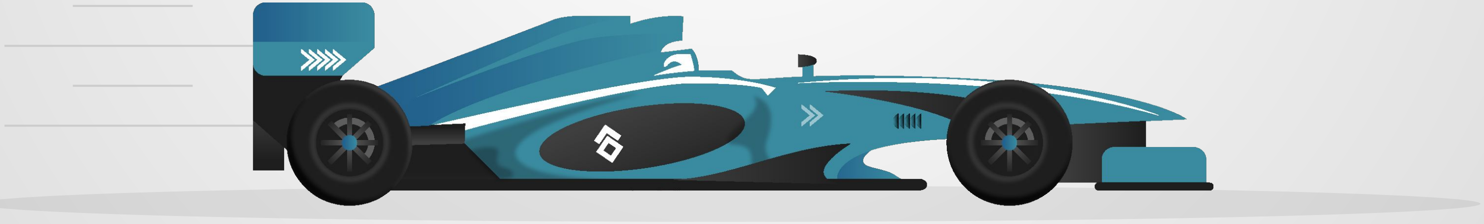
Quentin Phillips & Steven Jia



01

The Data

- Kaggle Dataset featuring tens of thousands of observations
- It contains data from every Formula 1 race from 1970 - 2023 (2024 data has not been entered yet)



Key Features

Final Position – Target Variable	• Finishing Position at the end of the race • If a racer was disqualified their finishing position is 20 - last
Circuit	• A number that represents each individual track • From 1 - 79
Weather	• 0 if Sunny, 1 if any inclement weather. Binary qualitative variable
Driver ID	• A number that represents each individual Driver • From 1 - 858
Team (Constructor)	• Numeric value to represent each individual team (Ferrari, Red Bull, Etc.)

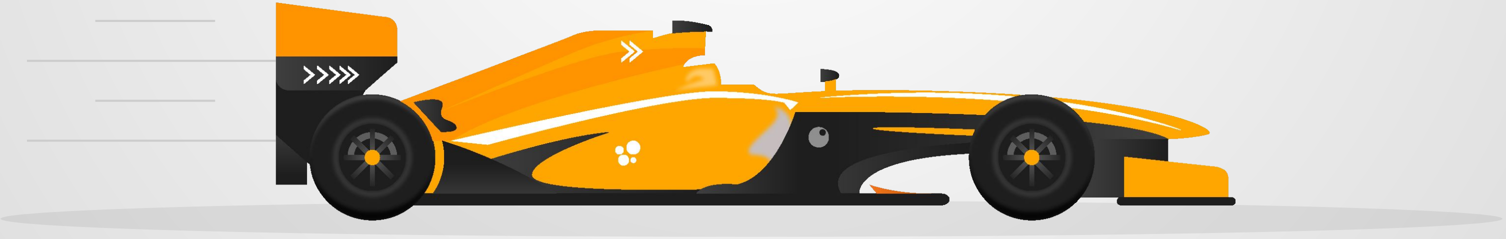
Preprocessing

Cleaned and prepared the data to be used in regression:

- Selected features
- Dropped null values
- Changed data format (time)
- Added Weather as a binary variable
- Restricted data to the last 5 years

02

Regression Analysis



Initial Linear Regression

- Our first model was Scikitlearn's LinearRegression
- Low R-squared
- High mean absolute error (MAE)
- x3 = Driver, x4 = constructor, x5 = qualifying position

```
Linear Regression Model - Train MAE: 8.077031458562976
Linear Regression Model - Test MAE: 8.760443668595483
```

OLS Regression Results

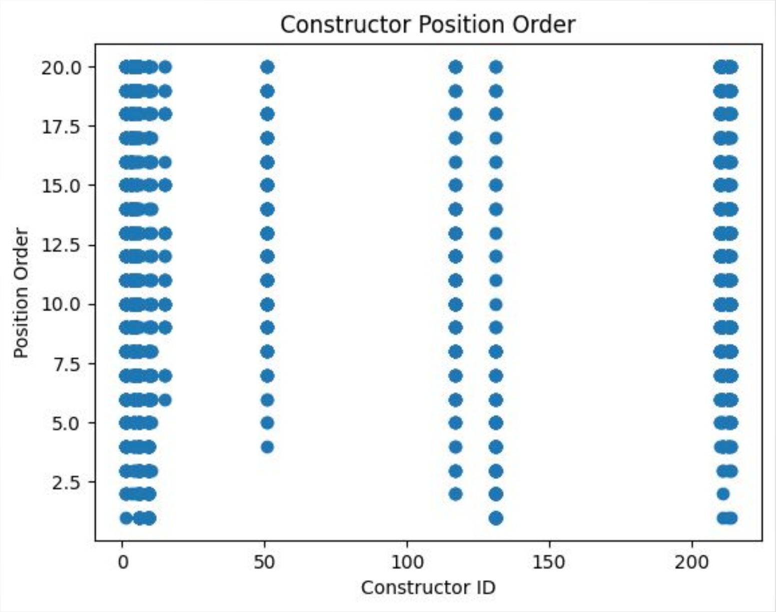
Dep. Variable:	y	**R-squared:**	0.385
Model:	OLS	**Adj. R-squared:**	0.384
Method:	Least Squares	**F-statistic:**	287.0
Date:	Mon, 06 May 2024	**Prob (F-statistic):**	8.10e-239
Time:	21:56:07	**Log-Likelihood:**	-6724.5
No. Observations:	2297	**AIC:**	1.346e+04
Df Residuals:	2291	**BIC:**	1.350e+04
Df Model:	5		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	3.1303	0.303	10.348	0.000	2.537	3.723
x1	-0.0006	0.004	-0.162	0.871	-0.008	0.006
x2	-0.0722	0.279	-0.259	0.796	-0.619	0.475
x3	0.0011	0.000	3.825	0.000	0.001	0.002
x4	0.0037	0.001	3.323	0.001	0.002	0.006
x5	0.6036	0.017	36.370	0.000	0.571	0.636

Omnibus:	237.441	**Durbin-Watson:**	1.027
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	317.530
Skew:	0.844	**Prob(JB):**	1.12e-69
Kurtosis:	3.683	**Cond. No.**	2.53e+03

Limitations of Linear Regression

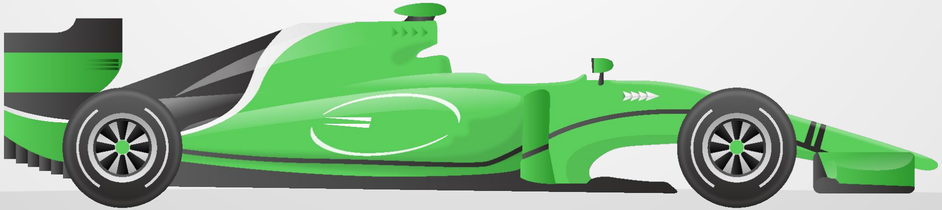
- Nominal Data
    - Driver ID, Constructor ID
- Lack of Model Complexity
- Noisy data



03

Neural Networks

•



Categorical Neural Network

- Approached the problem by defining the dependent variable as categorical
- Initial neural network to experiment with
- Test accuracy of 0.13

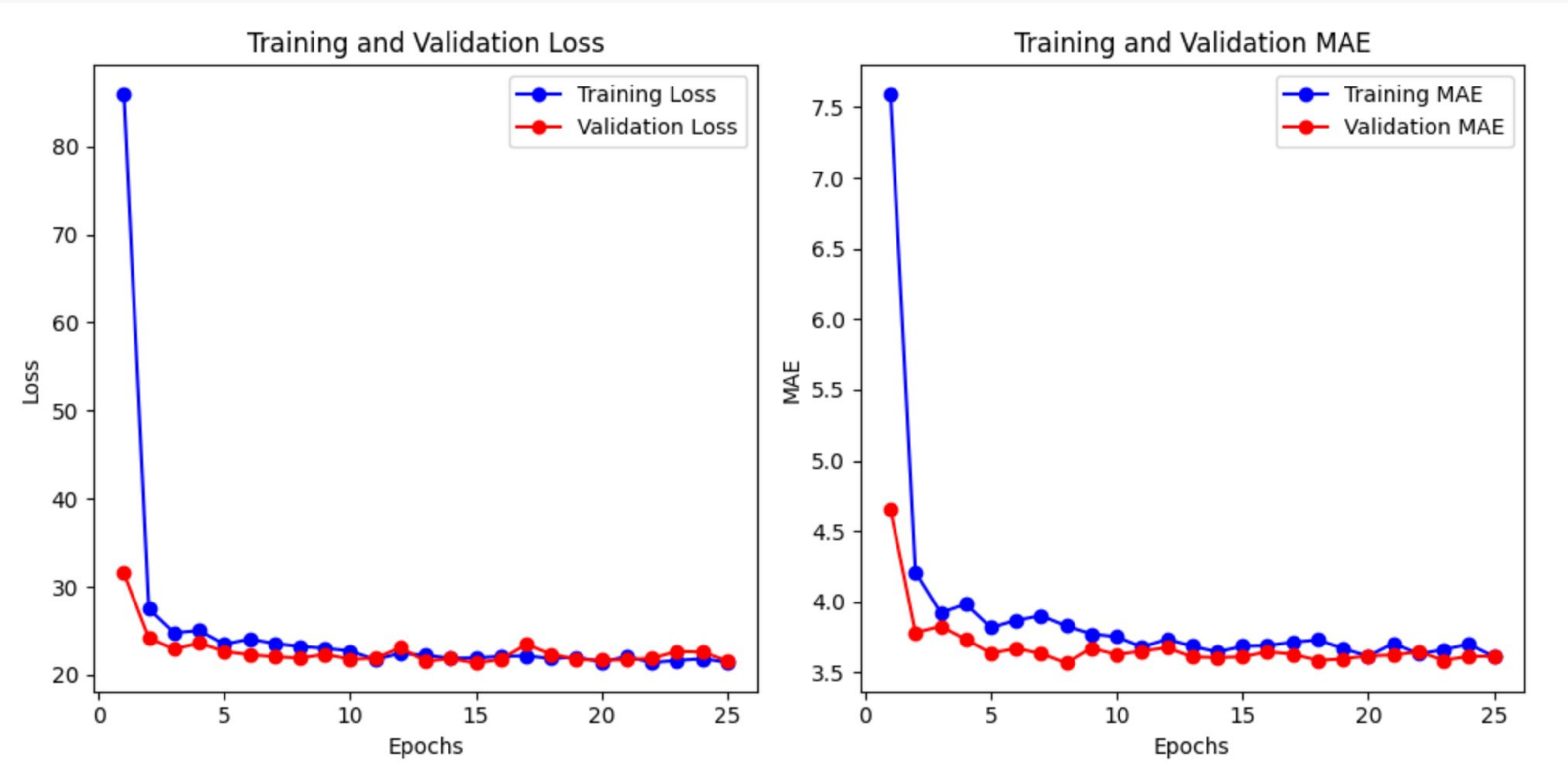
```
X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=0.2)

scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

model = Sequential([
    Dense(256, activation='relu', input_shape=(X_train.shape[1],)),
    Dropout(0.5),
    Dense(128, activation='relu', input_shape=(X_train.shape[1],)),
    Dropout(0.5),
    Dense(64, activation='relu'),
    Dense(12, activation='relu'),
    Dense(22, activation='relu'),
    Dense(34, activation='relu'),
    Dense(52, activation='relu'),
    Dense(82, activation='relu'),
    Dense(16, activation='relu'),
    Dense(21, activation='softmax')
])

optimizer = Adam(learning_rate=0.002)
model.compile(optimizer=optimizer, loss='sparse_categorical_crossentropy', metrics=['accuracy'])

early_stopping = EarlyStopping(monitor='val_loss', patience=20, restore_best_weights=True)
```

Linear Regression NN

- Switched to MSE loss function
- Measured MAE to determine performance
- Final testing MAE of 3.49
- Simple 5 layer network


```python
model_regression = Sequential([
    Dense(128, activation='relu', input_shape=(X_train.shape[1],)),
    Dropout(0.5),
    Dense(64, activation='relu'),
    Dense(32, activation='relu'),
    Dense(16, activation='relu'),
    Dense(1, activation='linear')
])

optimizer = Adam(learning_rate=0.001)
model_regression.compile(optimizer=optimizer, loss='mean_squared_error', metrics=['mean_absolute_error'])

early_stopping = EarlyStopping(monitor='val_loss', patience=10, restore_best_weights=True)

history = model_regression.fit(X_train_scaled, y_train, epochs=35, batch_size=32,
                    validation_split=0.2, callbacks=[early_stopping])

loss, mae = model_regression.evaluate(X_test_scaled, y_test)
print(f'Test Mean Absolute Error: {mae}')
```

Simple Linear MAE Plots

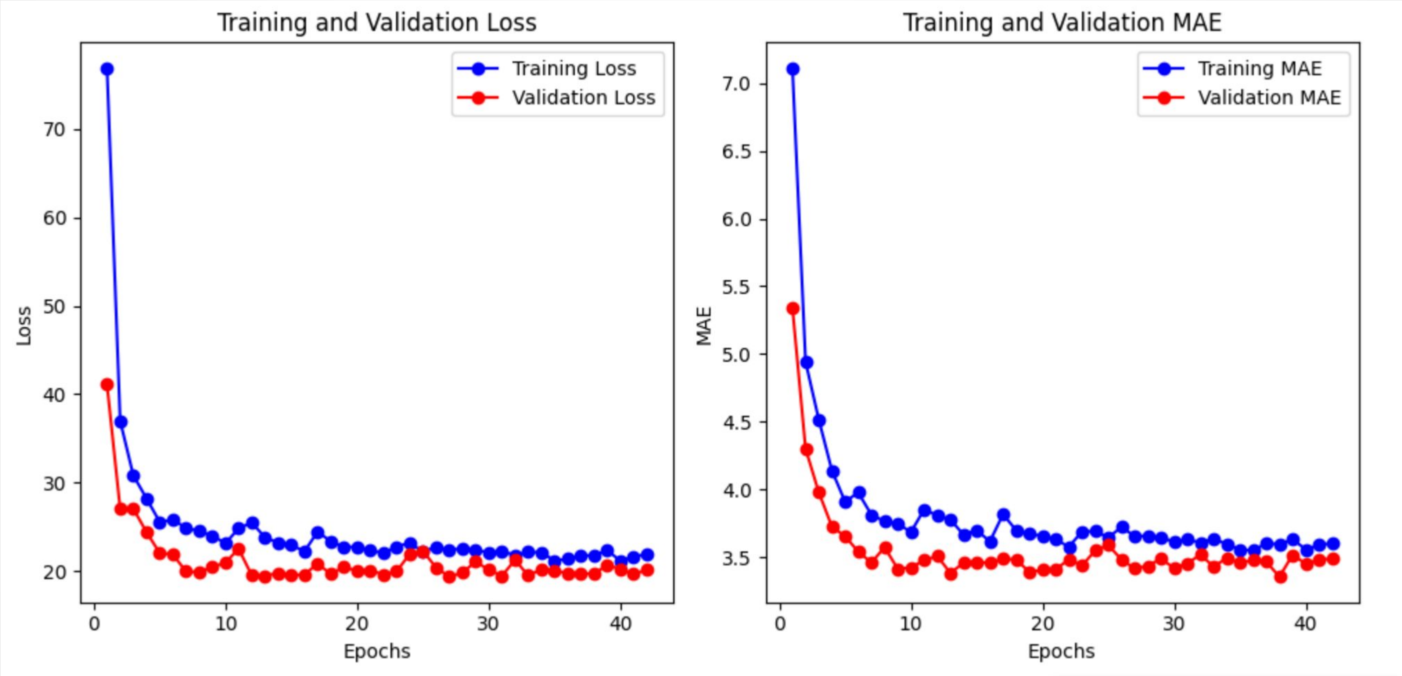


Linear Regression CNN

- Introduced the filters to the layers
- Adjusted hyperparameters to help CNN
- Final testing MAE of 3.54

```
from tensorflow.keras.layers import Conv1D, MaxPooling1D, Flatten

model_CNN = Sequential([
    Conv1D(filters=32, kernel_size=3, activation='relu', input_shape=(X_train.shape[1], 1)),
    MaxPooling1D(pool_size=2),
    Conv1D(filters=64, kernel_size=3, activation='relu'),
    MaxPooling1D(pool_size=2),
    Conv1D(filters=128, kernel_size=3, activation='relu'),
    MaxPooling1D(pool_size=1),
    Flatten(),
    Dense(64, activation='relu'),
    Dropout(0.45),
    Dense(128, activation='relu'),
    Dropout(0.45),
    Dense(1, activation='linear')
])

optimizer = Adam(learning_rate=0.001)
model_CNN.compile(optimizer=optimizer, loss='mean_squared_error', metrics=['mean_absolute_error'])

early_stopping = EarlyStopping(monitor='val_loss', patience=15, restore_best_weights=True)

history = model_CNN.fit(X_train_scaled.reshape(-1, X_train_scaled.shape[1], 1), y_train, epochs=100, batch_size=32,
                    validation_split=0.2, callbacks=[early_stopping])

loss, mae = model_CNN.evaluate(X_test_scaled.reshape(-1, X_test_scaled.shape[1], 1), y_test)
print(f'Test Mean Absolute Error: {mae}')
```

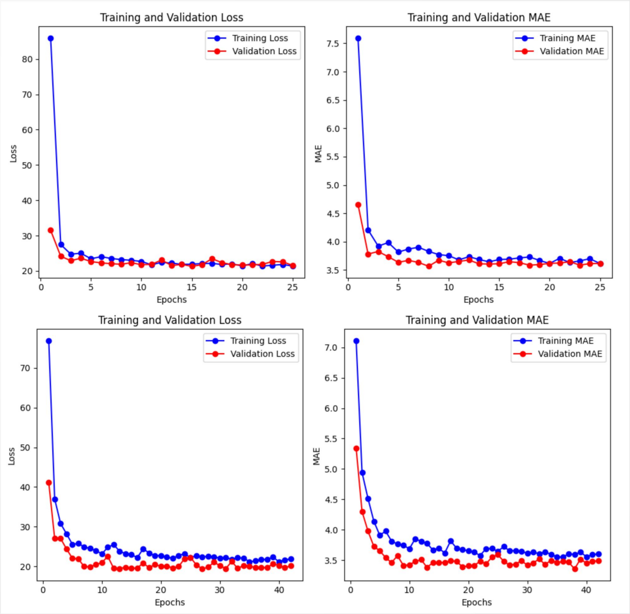
MAE Plots



Training and Validation Loss
Training Loss
Validation Loss
Loss
Epochs
70
60
50
40
30
20
0
10
20
30
40
Training and Validation MAE
Training MAE
Validation MAE
MAE
Epochs
7.0
6.5
6.0
5.5
5.0
4.5
4.0
3.5
0
10
20
30
40

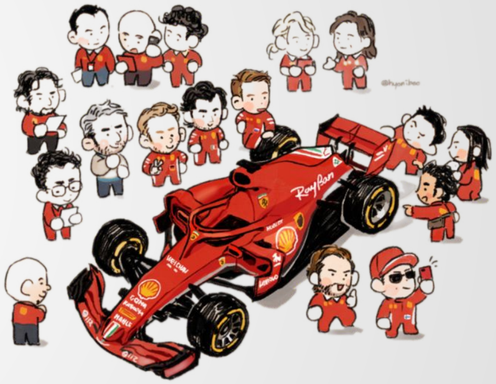
Comparing Neural Networks

- Neural networks had similar final MAE values
- There may be some room for improvement with the CNN based on the plots
- Future adjustments to the complexity of the CNN may be able to create a significant advantage over the initial neural network



Future Steps

- Improve complexity and output from both neural networks
- Attempt more complex regression models
- Predict and analyze results for future races



Thank You