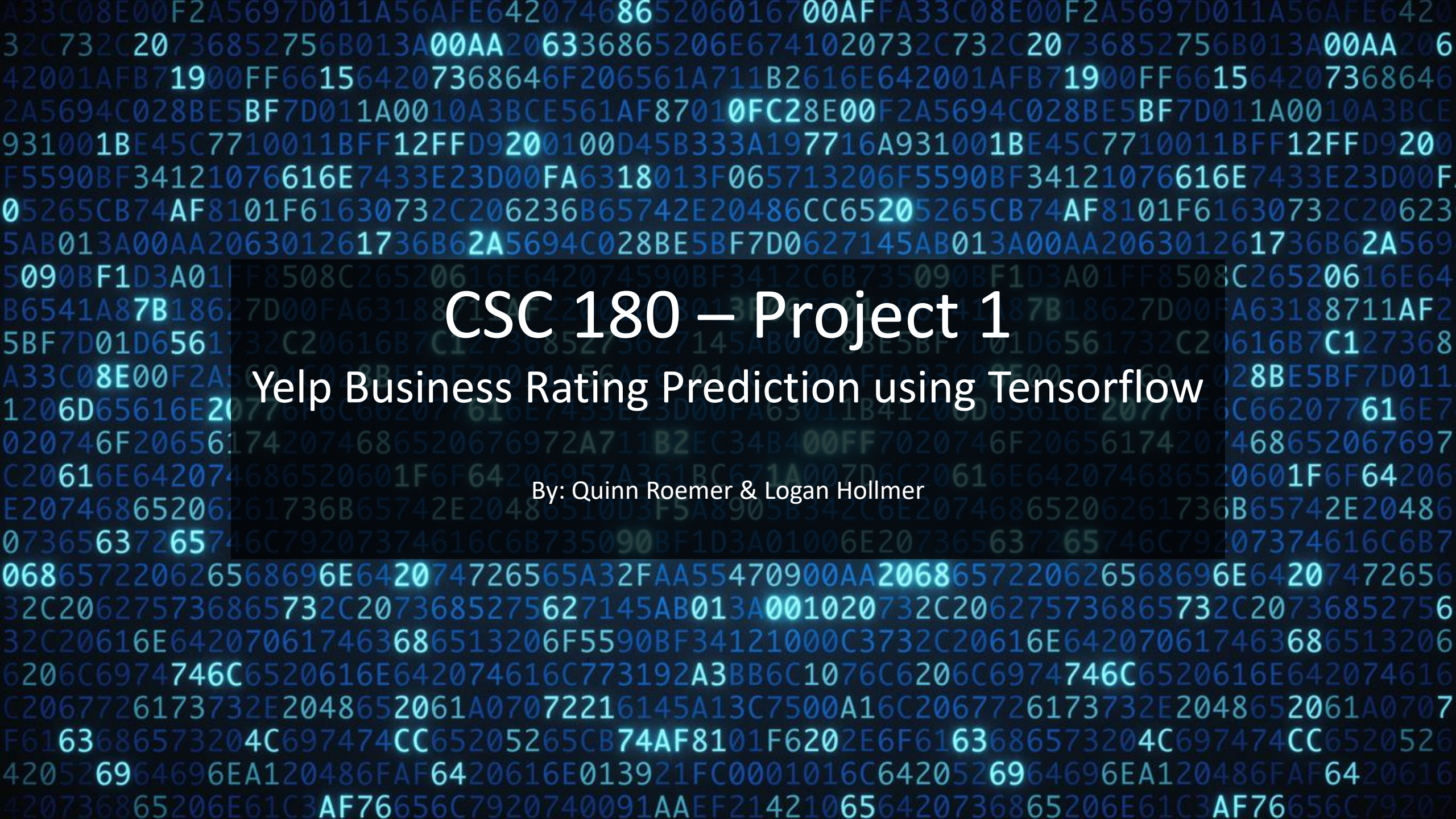


CSC 180 – Project 1

Yelp Business Rating Prediction using Tensorflow

By: Quinn Roemer & Logan Hollmer

Model/Code Design

```
#Perform TF-IDF analyse on data
vectorizer = sk_text.TfidfVectorizer(stop_words='english', max_features=2000, min_df=30, lowercase=True)
x = vectorizer.fit_transform(x)
print("X vectorized using TF-IDF...")
print(x)
print(x.shape)
print(len(vectorizer.get_feature_names()))
```

Quinn's Code

```
import sklearn.feature_extraction.text as sk_text
vectorizer = sk_text.TfidfVectorizer(
                                stop_words='english',
                                max_features = 1000,
                                min_df=1)
k=merged_inner['all_reviews']
matrix = vectorizer.fit_transform(k)
tfidf_data = matrix.toarray()
print(tfidf_data)
print(vectorizer.get_feature_names())
print(tfidf_data.shape)
```

Logan's Code

Code Differences

- max_features=1000 & max_features=2000
- min_df=1 & min_df=30

Model/Code Design

build_model()

- Between 0 and 5 hidden layers
- Between 10 and 200 neurons per layer
- Probability for high neuron count reduces for each additional hidden layer

```python
def activation_function():
  return random.choice(['relu', 'sigmoid', 'tanh'])

def build_model():
  model = Sequential()

  number_of_neurons=random.randint(191)+10
  model.add(Dense(number_of_neurons, input_dim=x.shape[1], activation=activation_function()))

  number_of_layers=random.randint(5)
  for i in range(number_of_layers):
    number_of_neurons=random.randint((number_of_layers-i)*20)+10

    model.add(Dense(number_of_neurons, activation=activation_function()))

  model.add(Dense(1))  # 1 output neuron

  model.compile(loss='mean_squared_error', optimizer=random.choice(['adam', 'sgd']))
  return model
```

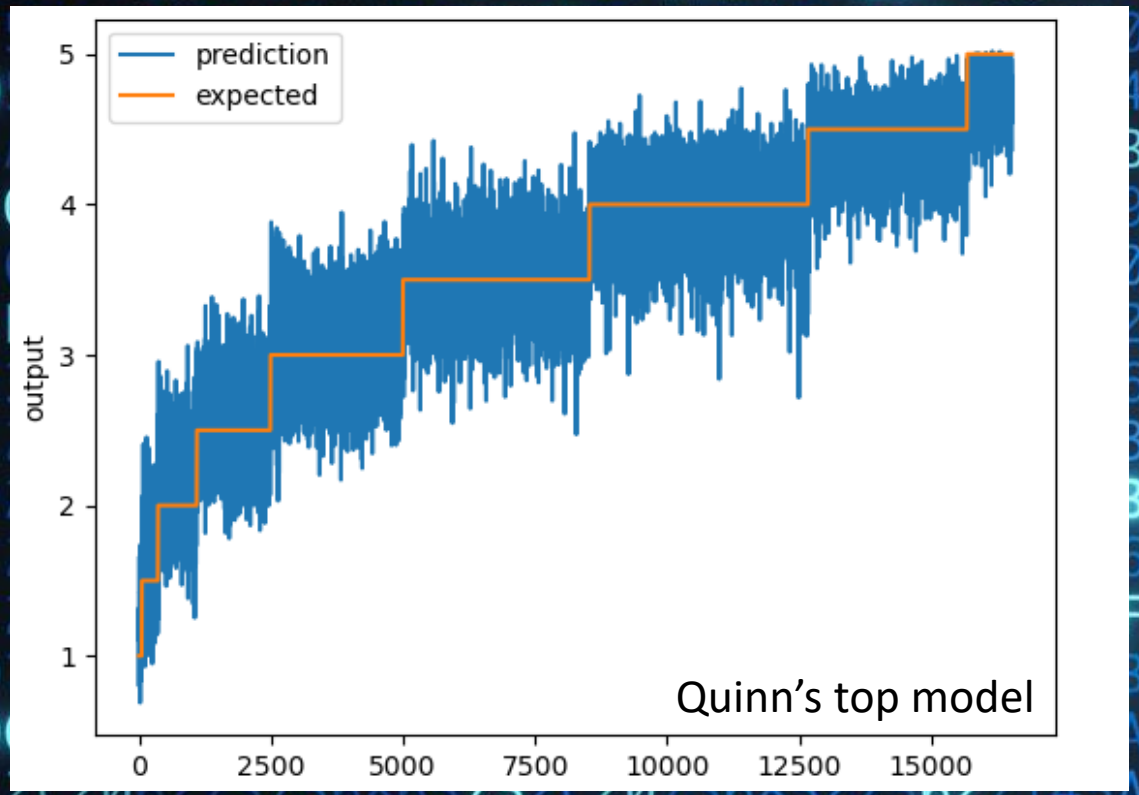
Logan's build_model() function

```python
from sklearn import preprocessing
import numpy as np

x = merged_inner['review_count'] #returns a numpy array
norm = np.linalg.norm(x)
normal_array = x/norm
```

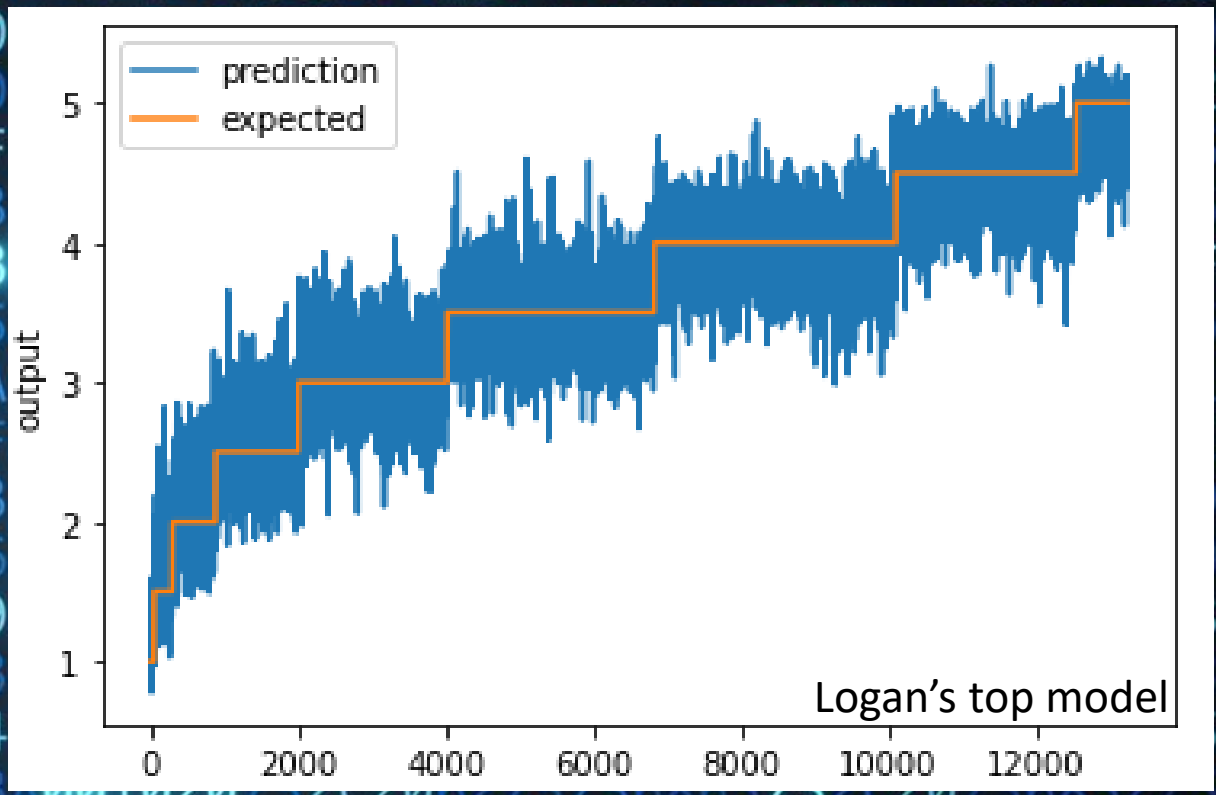
Linearized review_count

Findings/Results



Model Settings

- Hidden Layer 1: 96 neurons, tanh activation
- Hidden Layer 2: 13 neurons, tanh activation
- Hidden Layer 3: 15 neurons, tanh activation
- Optimizer: Adam
- RMSE: 0.2499



Model Settings

- Hidden Layer 1: 51 neurons, relu activation
- Optimizer: Adam
- RMSE: 0.2515

Findings/Results

Model ID	RMSE	Number of hidden Layers	Neurons per layer	Activation function	Optimizer
1	0.26034	1	15	relu	sgd
2	0.25064	3	38, 77, 16	tanh, tanh, tanh	adam
3	0.25053	3	95, 20, 70	tanh, tanh, tanh	adam
4	0.25031	2	66, 38	sigmoid, sigmoid	adam
5	0.24992	3	96, 13, 15	tanh, tanh, tanh	adam

Quinn's top 5 models in descending order

Task Division, Challenges Encountered, and what we Learned

Task Division

- Both group members created fully functional programs to pre-process data, vectorize data, and train models
- Logan created the majority of the report
- Quinn scripted and created the presentation

Challenges Encountered

- Memory issues due to data size, solved with frequent loading/unloading of data to disk
- Training time, solved by creating automatic scripts to train our models overnight

What we Learned

- More data is a good thing! extracting more features in the TF-IDF process would've probably created a better model
- Decreasing neuron counts in hidden layers following other layers is generally optimal
- Using the same activation function for all hidden layers is generally optimal