Lab 1 Assignment

Chase Hoffman

Introduction

A feedforward neural network was implemented to predict real estate prices. Data was taken from the UC Irvine machine learning repository and manipulated to work with the model. The model utilized the simple mean squared error loss function. Five data inputs were employed to generate a single output. This output was then verified with test data reserved from the set. Finally, linear regression was implemented to display the trend between the predicted model output and the actual outputs.

Discussion of Task

The first step in constructing the model starts with importing the necessary libraries for both data processing and building the model. The libraries utilized in this project were numpy, matplotlib, pandas, tf.keras, scipy and sklearn. After all libraries were imported, data preprocessing was performed to make the data usable for the model. Firstly, the columns that were not utilized were taken off of the data set using a function from the pandas library. Next, variables were created to split up the data into inputs and outputs. These two variables were then split into training and test data. 20% of the input data was reserved for testing data, while the other 80% was used to train the model. Normalization of input data at varying scale typically results in a more accurate model for tasks such as this one. Using functions from the sklearn library, all input values - both test and training data- were normalized to values between 0 and 1.

The model was then constructed. The model that was constructed contains one input layer, three hidden layers, and one output layer. The input layer consists of 5 neurons, the same number as the amount of inputs. All three hidden layers also consist of 5 neurons for simplicity. Through trial and error, it was decided that the “relu” activation function worked best for this model. The input layer as well as all three hidden layers employ this activation function. The “Adam” optimizer was chosen, as it seemed to perform quite well for this task. The loss function was the mean squared error loss function. After all this information was made explicit in the software, the model was run for 300 epochs. This number was chosen after much trial and error, and it seems to give the best results for the model. The model’s loss compared to the test data can be seen in figure 1. Further, the final value of loss given by the model is shown within figure 2 to be 46.78.

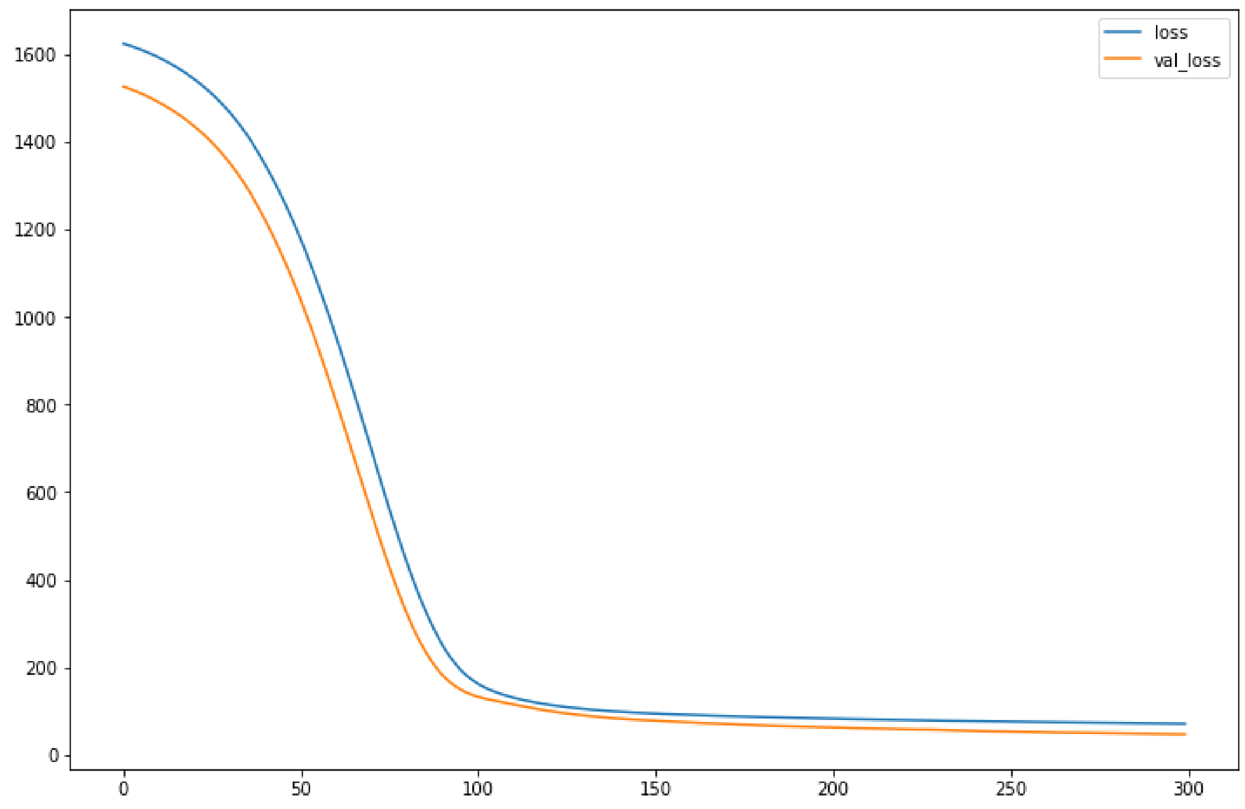
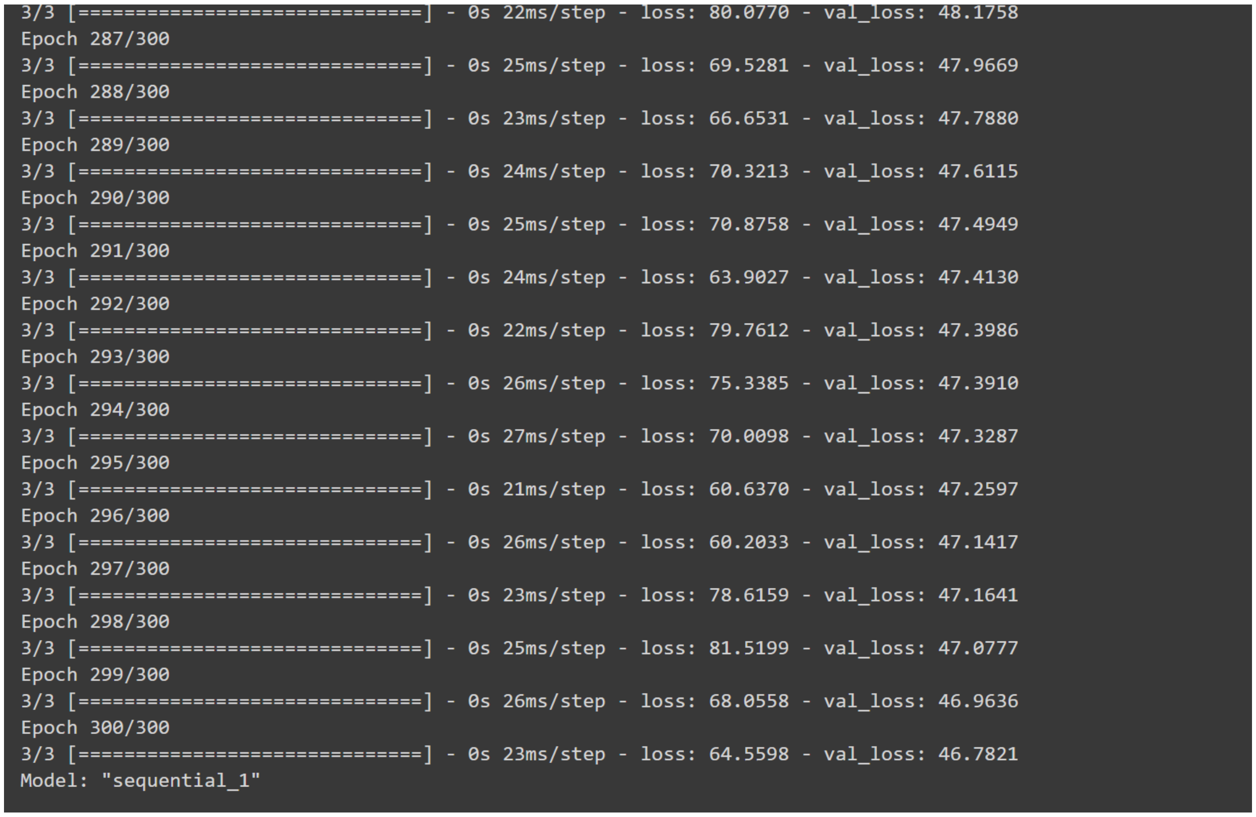


Figure 1: Model loss over 300 epochs



```
3/3 [==============================] - 0s 22ms/step - loss: 80.0770 - val_loss: 48.1758
Epoch 287/300
3/3 [==============================] - 0s 25ms/step - loss: 69.5281 - val_loss: 47.9669
Epoch 288/300
3/3 [==============================] - 0s 23ms/step - loss: 66.6531 - val_loss: 47.7880
Epoch 289/300
3/3 [==============================] - 0s 24ms/step - loss: 70.3213 - val_loss: 47.6115
Epoch 290/300
3/3 [==============================] - 0s 25ms/step - loss: 70.8758 - val_loss: 47.4949
Epoch 291/300
3/3 [==============================] - 0s 24ms/step - loss: 63.9027 - val_loss: 47.4130
Epoch 292/300
3/3 [==============================] - 0s 22ms/step - loss: 79.7612 - val_loss: 47.3986
Epoch 293/300
3/3 [==============================] - 0s 26ms/step - loss: 75.3385 - val_loss: 47.3910
Epoch 294/300
3/3 [==============================] - 0s 27ms/step - loss: 70.0098 - val_loss: 47.3287
Epoch 295/300
3/3 [==============================] - 0s 21ms/step - loss: 60.6370 - val_loss: 47.2597
Epoch 296/300
3/3 [==============================] - 0s 26ms/step - loss: 60.2033 - val_loss: 47.1417
Epoch 297/300
3/3 [==============================] - 0s 23ms/step - loss: 78.6159 - val_loss: 47.1641
Epoch 298/300
3/3 [==============================] - 0s 25ms/step - loss: 81.5199 - val_loss: 47.0777
Epoch 299/300
3/3 [==============================] - 0s 26ms/step - loss: 68.0558 - val_loss: 46.9636
Epoch 300/300
3/3 [==============================] - 0s 23ms/step - loss: 64.5598 - val_loss: 46.7821
Model: "sequential_1"
```

Figure 2: Output from model after 300 epochs

Following this, functions from the libraries matplotlib and sklearn were utilized to create a scatter plot of the actual output data compared to the predicted output data. This scatter plot, along with a regression line, can be seen in the below figure. The training points can be seen in blue while the testing points are in red, as instructed. As illustrated by figure 3, the model can predict the output with good accuracy.

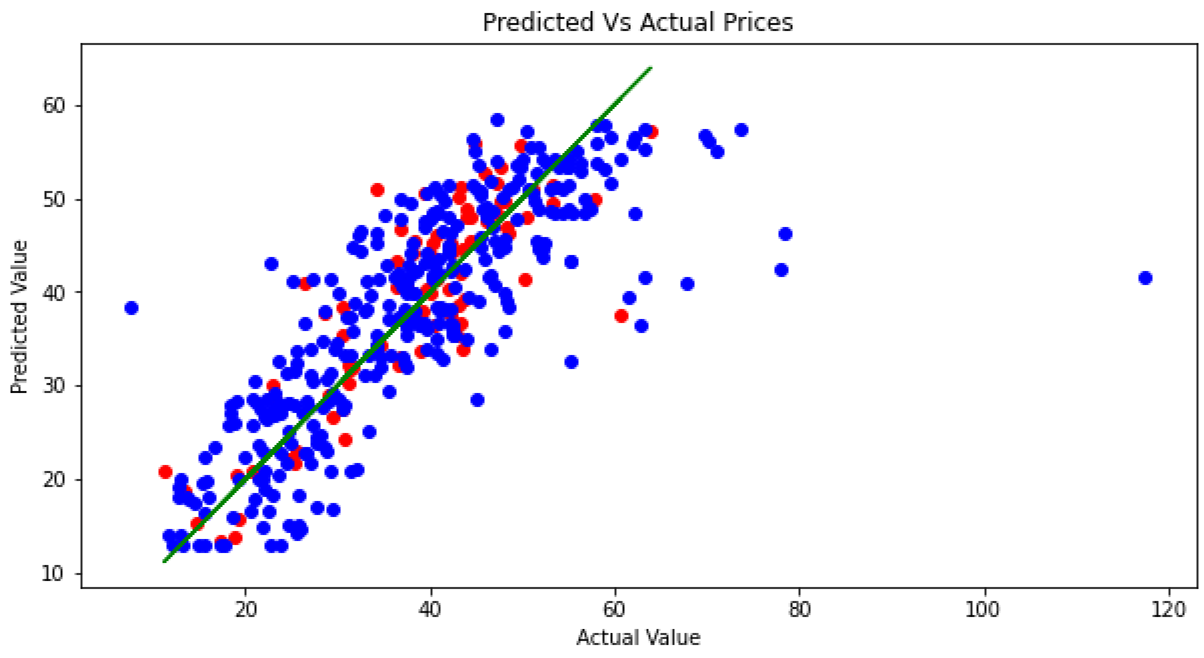


Figure 3: Predicted vs Actual values for data

Following the successful prediction of values from the model, linear regression was implemented. The data points on the scatter plot were used to find a slope and intercept for the cross-scatter could. This was done with a linear regression function from the python library scipy. This line was then plotted using matplotlib. The slope of the line was found to be .84. Furthermore, the intercept of the line was found to be 7.01. The actual line can be seen plotted in figure 4 below.

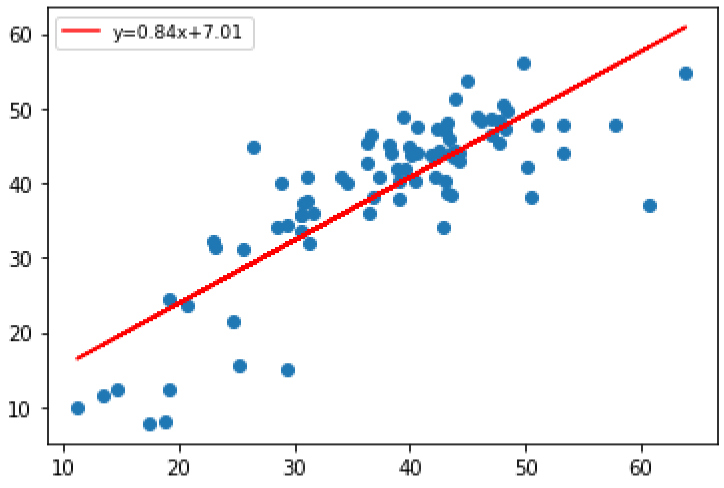


Figure 4: Linear regression plot of test data

Conclusion

Aspects of feedforward neural networks as well as mean squared error reduction were explored. Familiarization with machine learning software, such as keras, was gained. An understanding of what parameters are important in neural network design was also achieved. Implementation of linear regression techniques was accomplished through functions from python libraries.

Source code

```
import numpy as np

import matplotlib.pyplot as plt

from mpl_toolkits.mplot3d import Axes3D

import pandas as pd

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import Dense, Activation

from tensorflow.keras.optimizers import Adam


from keras.layers.core import Dense

from keras.layers import Input

from keras.models import Sequential

from keras import optimizers

from sklearn.model_selection import train_test_split


from google.colab import files

uploaded = files.upload()

import io

Data = pd.read_csv(io.BytesIO(uploaded['Real estate valuation data set
(1).csv']))

Data.head(7).T

Data = Data.drop('Unnamed: 8',axis=1)

Data = Data.drop('Unnamed: 9',axis=1)

Data = Data.drop('Unnamed: 10',axis=1)
Data = Data.drop('Unnamed: 11',axis=1)

Data = Data.drop('Unnamed: 12',axis=1)

Data = Data.drop('No',axis=1)

Data.info()

Data.describe().transpose()

X = Data.drop('Y house price of unit area',axis =1).values

y = Data['Y house price of unit area'].values

from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=101)

from sklearn.preprocessing import StandardScaler

s_scaler = StandardScaler()

X_train1 = s_scaler.fit_transform(X_train.astype(np.float))

X_test = s_scaler.transform(X_test.astype(np.float))

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import Dense, Activation

from tensorflow.keras.optimizers import Adam

model = Sequential()

model.add(Dense(5,activation='relu'))

model.add(Dense(5,activation='relu'))

model.add(Dense(5,activation='relu'))

model.add(Dense(5,activation='relu'))

model.add(Dense(1))

model.compile(optimizer='Adam',loss='mse')
model.fit(x=X_train1,y=y_train,

          validation_data=(X_test,y_test),

          batch_size=128,epochs=300)

model.summary()
```

```
loss_df = pd.DataFrame(model.history.history)

loss_df.plot(figsize=(12,8))
```

```
y_pred = model.predict(X_test)

y_pred_train = model.predict(X_train1)

fig = plt.figure(figsize=(10,5))

plt.scatter(y_test,y_pred, color = "red")

plt.scatter(y_train,y_pred_train, color ='blue')

plt.title("Predicted Vs Actual Prices")

plt.ylabel("Predicted Value")

plt.xlabel("Actual Value")
```

```
from scipy import stats
z = pred_output.reshape(414,)
slope, intercept, r_value, p_value, std_err =
stats.linregress(real_output,z)
line = slope*real_output+intercept
plt.plot(real_output, line, 'r',
label='y={:.2f}x+{:.2f}'.format(slope,intercept))
plt.legend(fontsize=9)
plt.scatter(real_output,z)
plt.show()
```