CS4287 - Project 1

Linear Regression, Ronan McMullen, 0451657

Linear Regression

Model!

Linear regression seeks to model the relationship between dependent variables and a single independent variable.

This model can be used to predict the dependent variable based on an instance of independent variables.

Data!

I have chosen to use the Ames Housing Data Set for this project.

It is a large data set with over 1400 entries. Each entry is composed of over 70 attributes.

www.kaggle.com/c/house-prices-advanced-regression-techniques

Implementation!

Using Keras and other python based machine learning libraries I plan to implement a dense neural network that can display proof that it is learning to predict house prices based off of input from the Ames data set.

Data Cleaning

In an attempt to maximise the accuracy of the neural network some data cleaning was required.

- Columns and rows with large amount of null values were dropped.
- Any rows with a null dependent variable (SalePrice) were dropped.
- Any remaining null values were assigned the value of 0.
- ID column was dropped as it does not contribute toward the value SalePrice.
- Entries with “Abnormal” entries in SaleCondition were dropped.

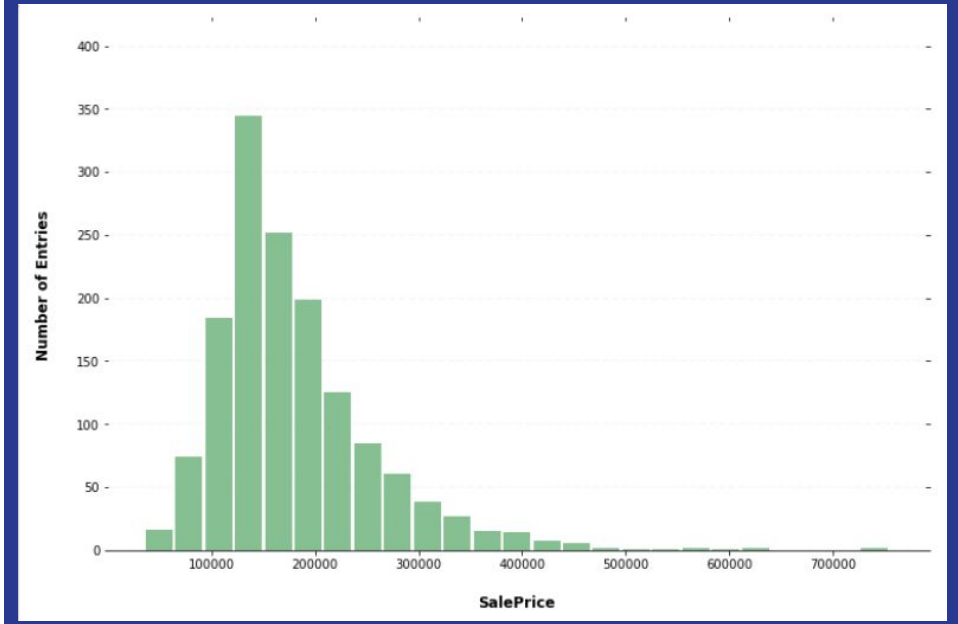
```
df.dropna(1, thresh=((df.shape[0]) * 0.5), inplace=True)
df.dropna(0, thresh=(df.shape[1] * 0.5), inplace=True)
df.replace(np.nan,0, inplace = True)
df.dropna(subset = ["SalePrice"], inplace = True)
df.drop(['Id'], axis=1, inplace = True)
abnormalSales = df[df.SaleCondition == "Abnorml"]
df.drop(abnormalSales.index, inplace = True)
```

To drop or not to drop?

You can see to the right that while the majority of Sale Prices fall between 100,000 and 300,000 there are some outlying values up around the 500,000+ range. This outlying data resides at the edges of the distribution and so is not reflective of the main body of the data.

With this in mind it was tempting to drop the rows containing these values as doing so could lead to an increase in statistical significance. However, they are natural members of the dataset and so must be included if the model is to be realistic in it's ambitions.

*Values that are more than 3*std away from the mean Sale Price can be considered outliers.*



SalePrice has a skewed distribution.

Data Preprocessing

Before using the data it was necessary to apply some appropriate preprocessing techniques.

- Data was split into a training and a testing set. I chose a train/test split ratio of 80/20.
- Dependent variable was scaled for validation purposes.

```
rnd.seed()
(train, test) = train_test_split(df, test_size=0.20, random_state=rnd.randint(1,
maxSale = train["SalePrice"].max()
trainY = train["SalePrice"] / maxSale
testY = test["SalePrice"] / maxSale
```

Data Preprocessing

- One Hot encoding was applied to the categorical data in each set. After encoding, the first column of each resulting One Hot vector was dropped as it is superfluous.
- The continuous data was fitted to the training set and then normalised across both training and testing data. Values in training will span from 0-1, values in test may be greater than one.
- At this point in the notebook all* data lies between 0 and 1.

except of course those continuous values within the test data set that were larger than the largest value from the corresponding feature in the training set.

```
lb = LabelBinarizer()

trainCat = np.array([]).reshape(len(train), 0)
testCat = np.array([]).reshape(len(test), 0)

for i in catCol:
    #talk about asType(str) here
    lb.fit(df[i].astype(str))

    temp = lb.transform(train[i].astype(str))
    trainCat = np.hstack([trainCat, temp[:,1:]])

    temp = lb.transform(test[i].astype(str))
    testCat = np.hstack([testCat, temp[:,1:]])

print(trainCat.shape)
mms = MinMaxScaler()
#talk about fit_Transform vs transform
trainCon = mms.fit_transform(train[conCol])
testCon = mms.transform(test[conCol])

trainX = np.hstack([trainCat, trainCon])
testX = np.hstack([testCat, testCon])
```

The Model

The following are some attributes of my model.

- **Sequential**: Each layer of the model is defined separately. A step by step process. Within Keras you have to define a Sequential object and then add each layer definition to it.

- **Densely connected**: **Each** node in a layer is connected to **every** node in the surrounding layers. Fully connected.

- **Multi Layered Perceptron**

The Model

- **RELU Activation:** The nodes in the input and hidden layers in the model have a rectified linear activation function applied to them. These nodes will either output the positive input the recieve or zero.

- **Single, Linear Output Node**

- **Adam Optimiser:** Adam applies a type stochastic gradient descent in an attempt to minimise the loss function of the model. Adam is regarded as being lightweight so it is an appropriate choice for this project.

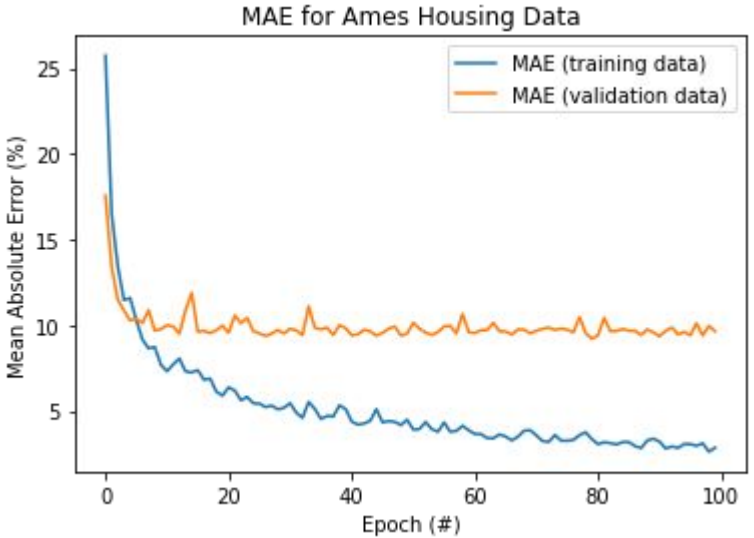
- **Mean Absolute Percentage Error Loss Function**

The Model

```
model = Sequential()

model.add(Dense(16, input_dim=trainX.shape[1], activation="relu"))

model.add(Dense(32, activation="relu"))
model.add(Dense(56, activation="relu"))
model.add(Dense(32, activation="relu"))
model.add(Dense(16, activation="relu"))
model.add(Dense(8, activation= "relu"))

model.add(Dense(1, activation="linear"))

opt = Adam(lr=1e-3, decay=1e-3 / 200)

model.compile(loss="mean_absolute_percentage_error", optimizer=opt)
```

Performance

I feel that overall my model performed reasonably well. The below plot illustrates a well performing run. As you can see there is quite a difference between the best and worst predictions which I am attributing to outlying data members.




```
Most accurate prediction: 99.98%
Least accurate prediction: 2.25%
Average accuracy: 90.35%
Mean Error: 9.65%, std: 9.93%
```

Conclusion

I *really* enjoyed this project. A neural net implementation is something I've always wanted to get under my belt. This was also the first project I have completed using Python.

I initially used a different and less expansive dataset for my model but I was uninspired with the first iteration of the project. I think the medical expense data I chose was already quite sanitised and so required little in the way of cleaning.

The Ames set provided a nice challenge and I feel that it encouraged me to dive a little deeper into analysing the original data file.

Looking forward to getting stuck into the next project now!

Ronan