AI Final Project – Black Friday Sales Prediction Analysis

Objective:

A retail store that specializes in three major categories of products collects numerous transactions from customers and intends to investigate on what factors may influence the purchase of customers the most, which essentially makes up their revenue. They would like to know customers’ consumption behavior mainly based on their demographics and shopping records to discover insights that could boost their sales.

The goal of my analysis is to predict the dependent variables in column 12 which is the amount of purchase for each transaction with the assistance of the information contained in the other independent variables. The amount of purchase is a continuous variable instead of categorical variable, thus we are facing a regression problem. Additionally, it is a supervised learning task since it is given labeled data. In this case, we are building models and training models by splitting the data into training, and testing data sets. Then I plan to perform a grid search to do the hyperparameter optimization to find out the best model. The end goal is to come up an ANN regression model with high accuracy to predict how much the customer is going to spend on black Friday given certain attributes such as their demographic attributes and transaction history.

Data source:

The website is <https://www.kaggle.com/mehdidag/black-friday>

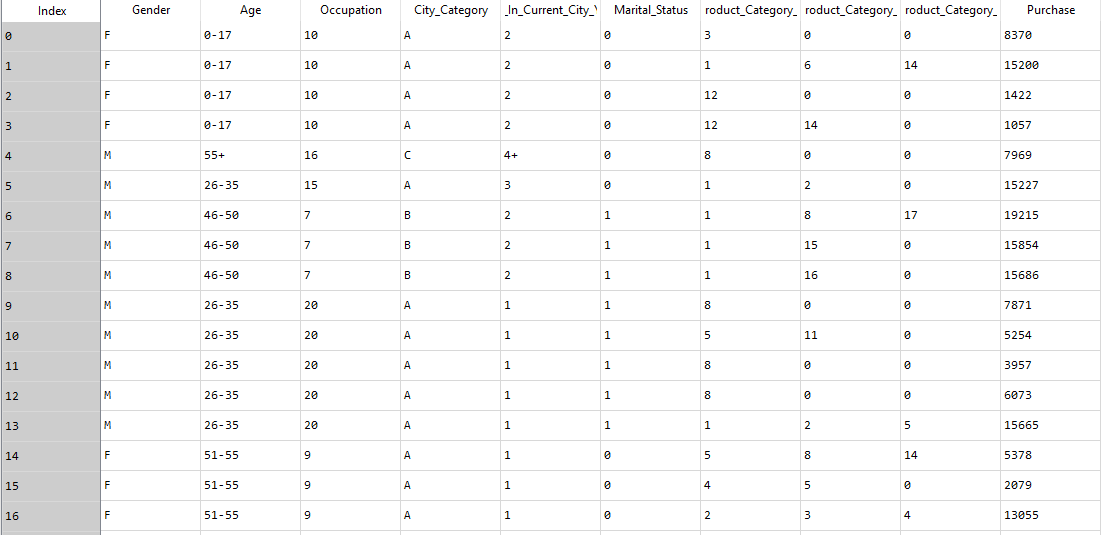
Data characteristics:

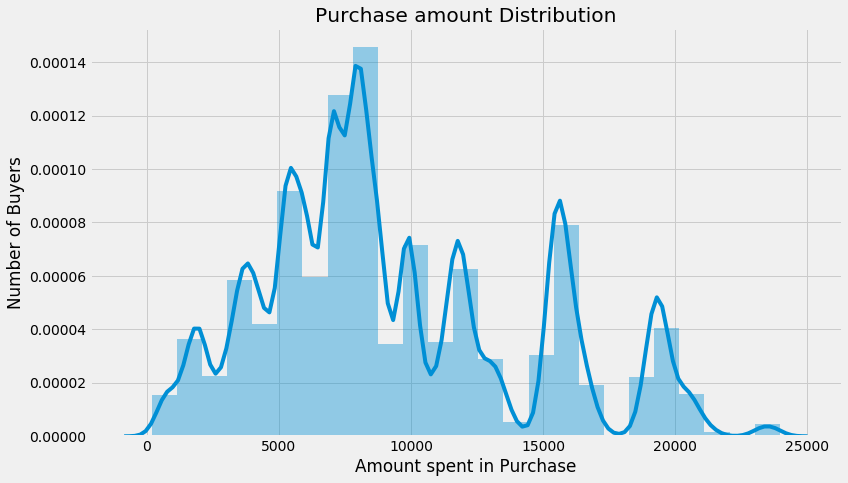
The data set consists of 537878 records and 12 columns. Purchase is how much a customer spends in a single transaction we collect. It is the variable we are trying to predict. Three product categories which indicate what products customers purchased in the store are categorical variables with missing data in two of them. City \_category and occupation in column 5 and 6 are also categorical variables which indicate in which city the transaction was conducted and the occupation of the person who conducted the transaction. Marital status is a binary variable of which 1 represents married and 0 represents single. Gender is a categorical variable therefore requires transformation that will be elaborated in our latter analysis. Additionally, the description of items in each product category and city categories are masked and replace with numbers in the data. From the analysis perspective, we are unable to analyze what each number corresponds to in the product categories. As the store, we aim to discover significant characteristics related to customers’ purchase behavior so that we are able to target them better with corresponding marketing strategies. The reason I choose this dataset is because it aims to solve a real-world business problem.

My analysis will be broken into five steps which are EDA, data processing with feature engineering, data partition, model building and conclusion.

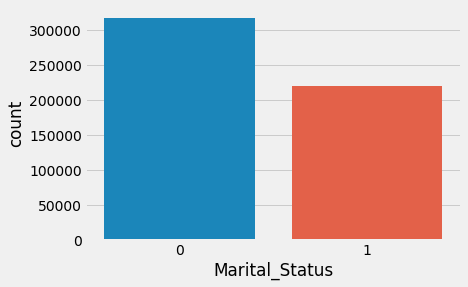
Exploratory Data Analysis:

The goal of EDA analysis is to assess each predictor and gain an acquaintance of the data. We are analyzing if there is any trends or patterns by visualization tools. The picture below is what dataset looks like.

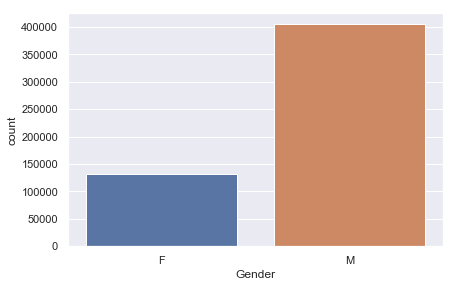




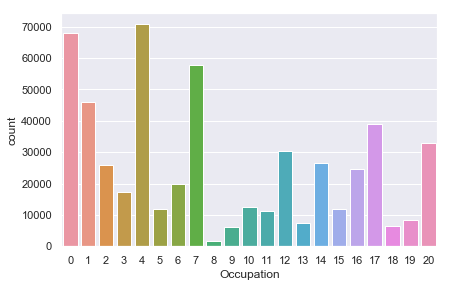
It seems like that our target variable has an almost normal distribution. The majority of population spends in range of 5000 to 10000 in the store.

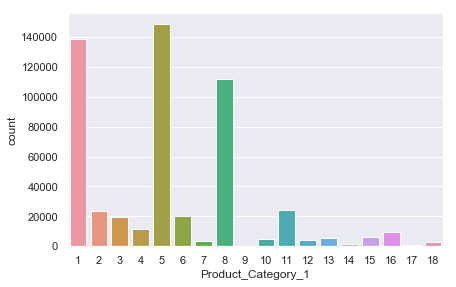


Apparently from the graph shown above, single people tend to buy more products, having more transactions than married people. However, there is no clear indication that single people spend more in their purchase than married people. I speculate that married people might spend more money than single people.

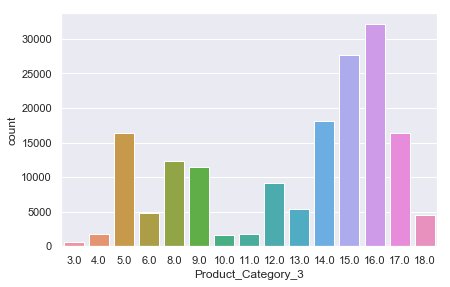
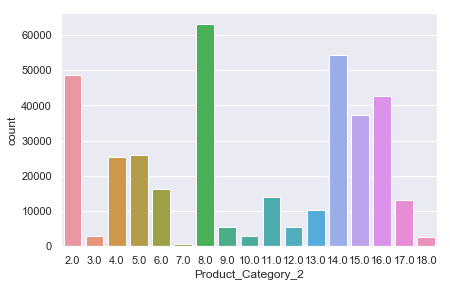


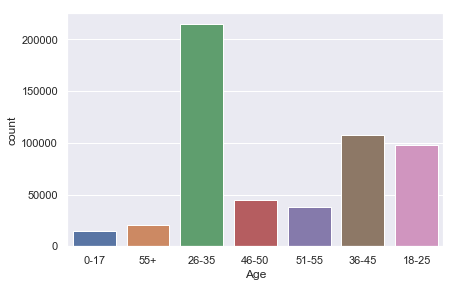
It is surprising that male has higher records of transactions than male.



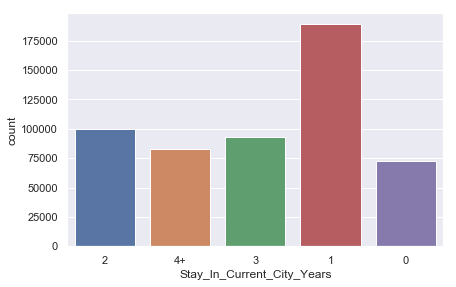


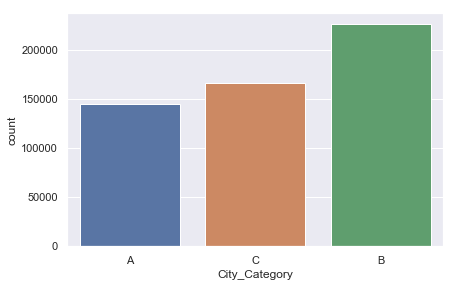
As indicated from pictures above, occupation has at least 20 different value which suggest that the store carries a wide variety of stocks geared towards public groups regardless of professions. In this dataset, we don’t know what each occupation number corresponds to therefore it is difficult to make any more analysis. It is the same case for three different product categories shown above and below.



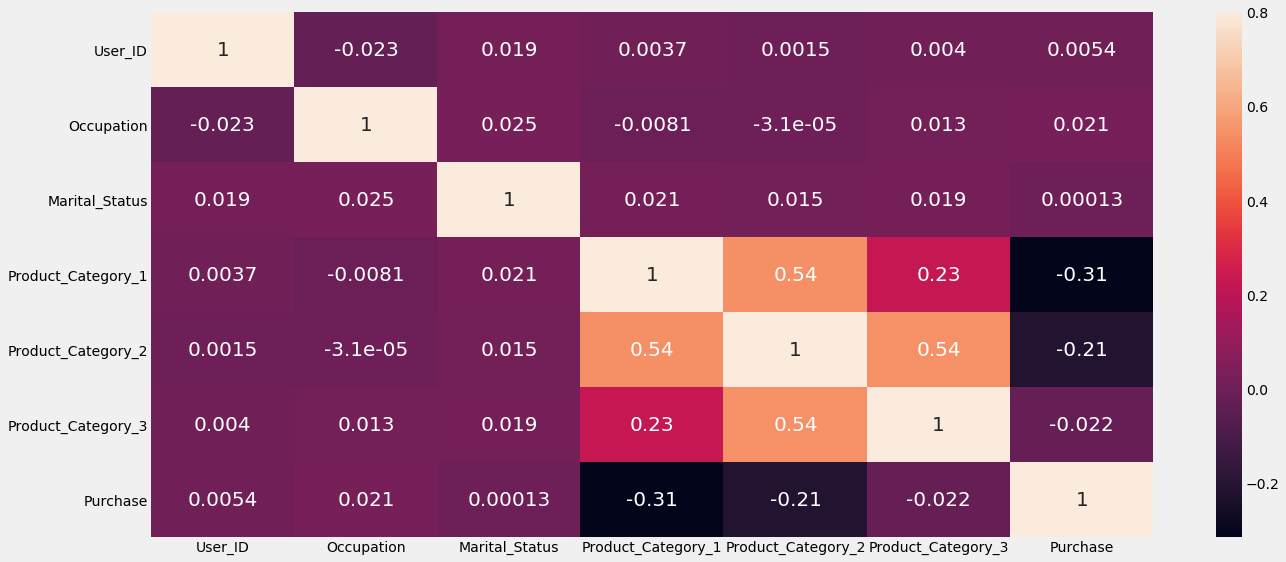


As expected, most purchases are made by people between 26-35.





I draw a correlation table for each variable. On the diagonal, each variable is perfectly correlated with itself. However, we observe that the correlations between different variables other than itself is relatively weak which is good for our analysis because there does not exist collinearity between variables. Collinearity may bias our model.



Preprocessing, feature engineering and data partition:

At first, I drop the first two columns User\_ID and Product\_ID which are sequences of number that are not significant to our analysis. Second, I use fillna function to replace the NaN with 0 in the product\_category2 and 3. Then I count the number of different items in each column and find out that there are 20 types of products including the 0s’ category which is replaced with NaN earlier.

In the feature engineering section, I convert gender from string to binary and age to numeric values. I create a dictionary to substitute age bins with numbers from 0 to 6. I also convert City\_category to binary from 0 to 2. For Stay\_in\_Current\_City\_Years, I create 5 dummy variables for this column. Hence, our columns increase.

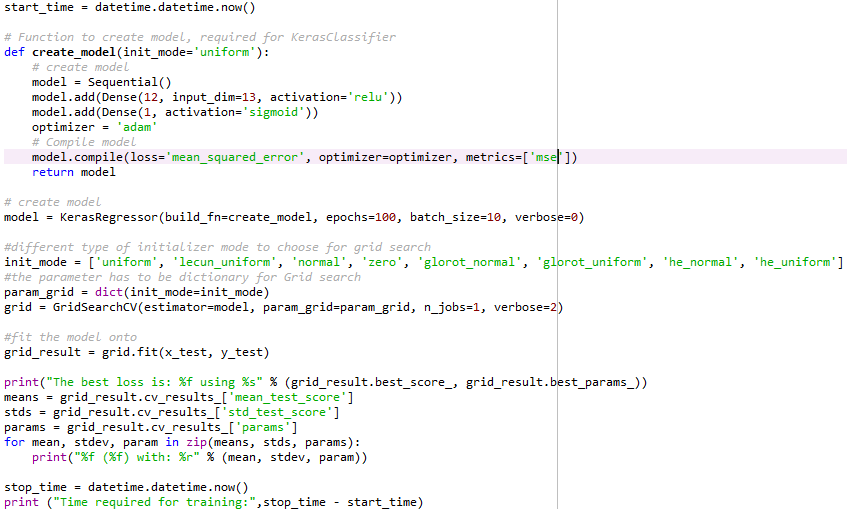
Before I build the model, I standardize x and scale y. Next I split the data set into x and y in which each represents response variables and target variable. For each x and y, I shuffle the dataset and take 5000 rows of the original dataset. Then I split them into 80% of training and 20% of testing.

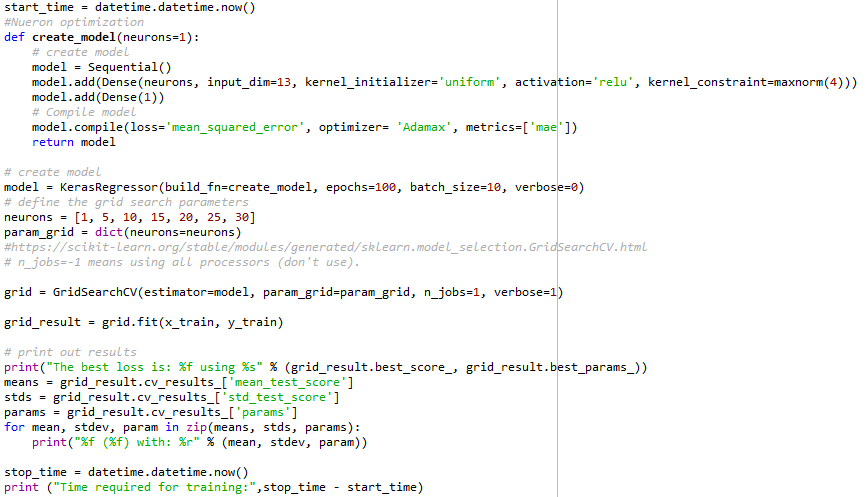
Model building and experimentation:

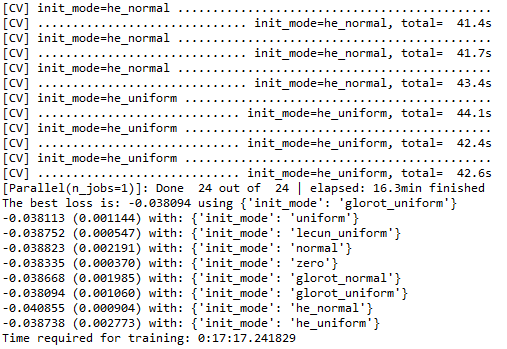
To build the traditional ANN models, I choose to build 2 dense layers in sequential order by using the feedforward network. I choose MSE as my loss and metrics because it is in regression setting. A metric is a function that is used to judge the performance of my model and it is similar to loss function.

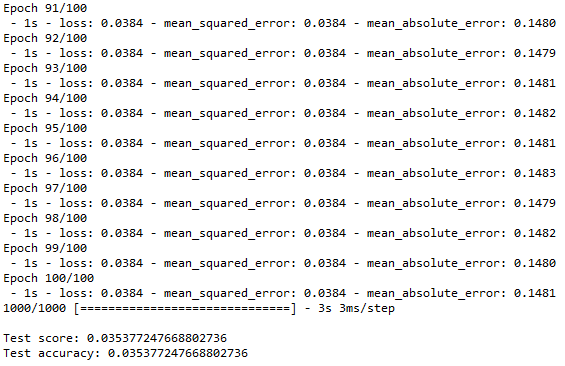
The first thing I tried was to experiment with Grid Search to search for the best hyperparameter that optimize the network. The result for grid search in init-mode was glorot-uniform that yields the lowest loss. However, I also attempted it with other hyperparameters such as neurons in the dense layers. It took me huge amount of time for the algorithm to run. Therefore, I switch my method by doing a for loop for different number of neuron in the dense layer.

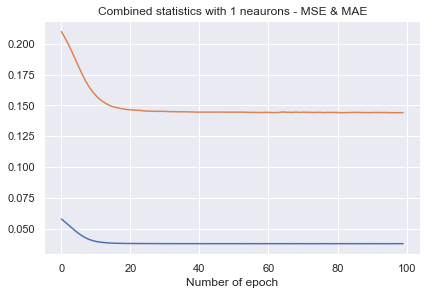
I did not time exactly how long it took to run all the models. But I estimate it took me 3 hours at least to run all models.

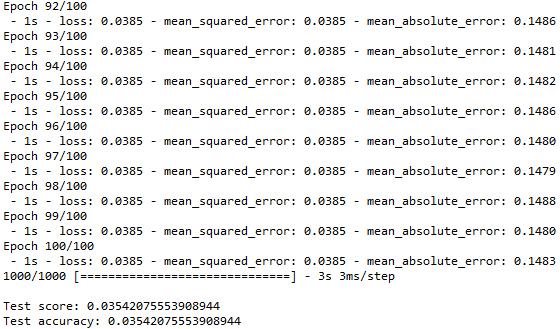


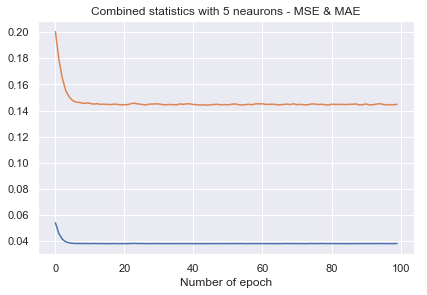


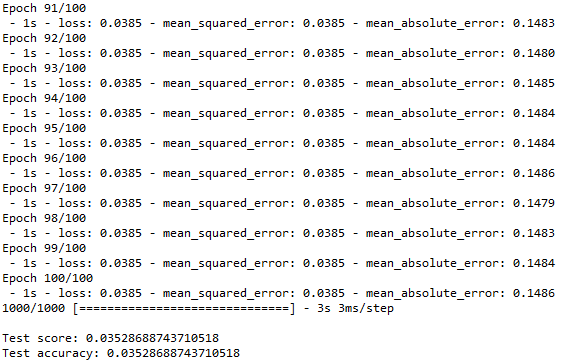


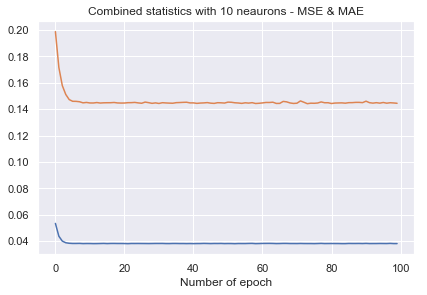


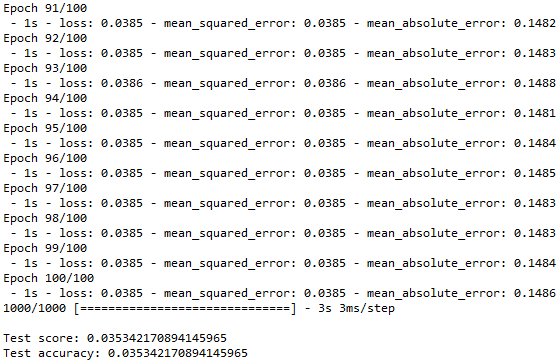


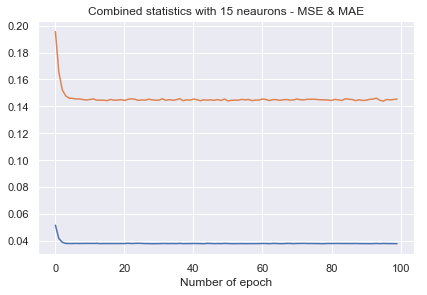


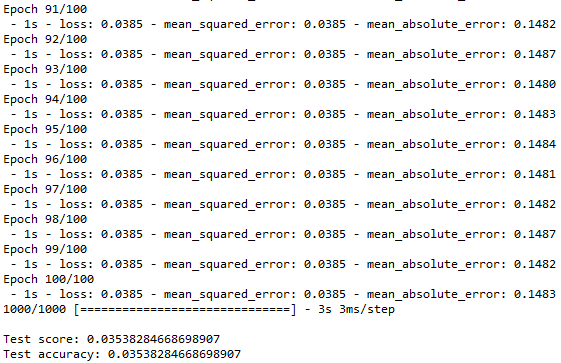


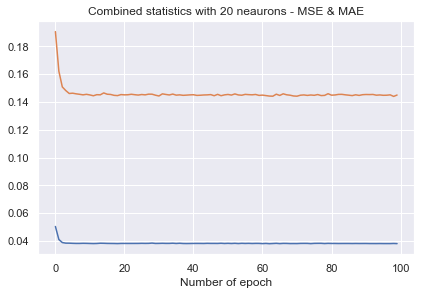


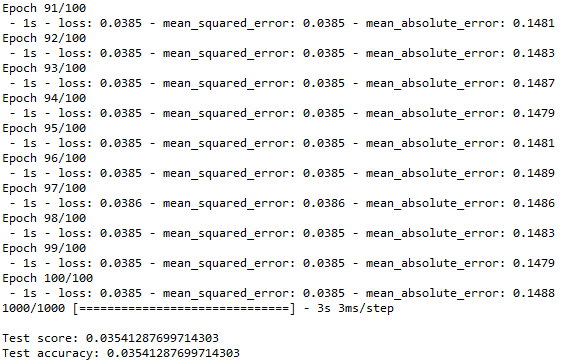


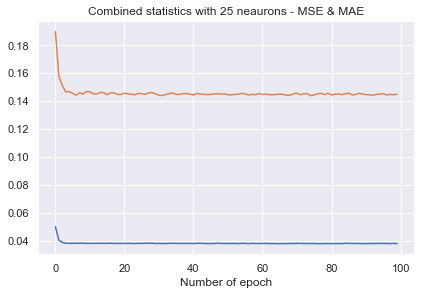


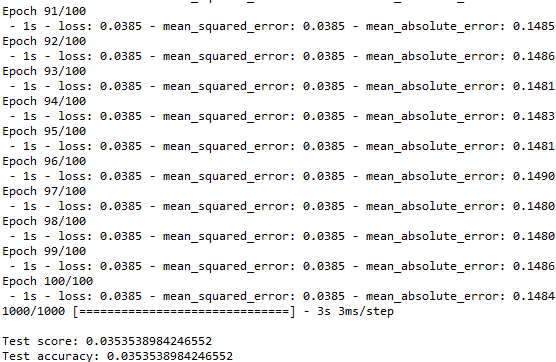


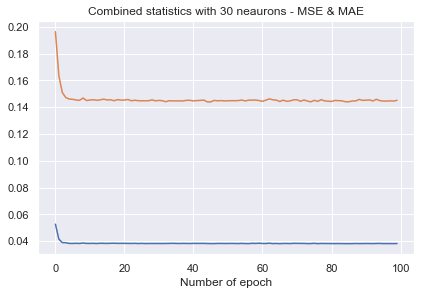












In conclusion, I would say that the model with 10 neuron has the best accuracy compared with others. As we can see, the mse goes down as we run more epochs. I used relu, sigmoid as the activation function for dense layers because they usually work well with regression setting. It is a supervised learning task and 80 20 data partition. I did not experiment with number of hidden layers, learning rates and momentum because we experimented them in the previously assignment and it would take a long time to run. I have learned a lot by fiddling with parameters this semester. The best way to learn one thing is by actually doing it along with being able to explain the question and solution in a laymen’s term.

Thank you!!

Code Appendix

"""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""

Import Libraries Section

"""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""

import numpy as np

import pandas as pd

import seaborn as sns

import matplotlib.pyplot as plt

from keras.models import Sequential

from keras.layers import Dense

from sklearn.preprocessing import StandardScaler

from sklearn import model\_selection

from sklearn import preprocessing

from matplotlib import pyplot

from sklearn.preprocessing import LabelEncoder

from keras.wrappers.scikit\_learn import KerasRegressor

from sklearn.model\_selection import GridSearchCV

import datetime

from keras.constraints import maxnorm

"""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""

Load Data Section

"""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""

pd.options.display.max\_columns = 999 #to visualize the whole grid

#read in the csv file into a dataframe

dataframe = pd.read\_csv("BlackFriday.csv",

delimiter = ",", header =0)

#,error\_bad\_lines=False

print(dataframe.count()) #count the number of rows for each row

dataframe.head()

dataframe.info()

dataframe.describe() #gives statistical value for each column

#check for duplicates:

idsUnique = len(set(dataframe.User\_ID))

idsTotal = dataframe.shape[0]

idsDupli = idsTotal - idsUnique

print('There are ' + str(idsDupli) + ' duplicate IDs for ' + str(idsTotal) + ' total entries')

"""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""

Data Visualization

"""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""

plt.style.use('fivethirtyeight')

plt.figure(figsize=(12,7))

sns.distplot(dataframe.Purchase, bins = 25)

plt.xlabel('Amount spent in Purchase')

plt.ylabel('Number of Buyers')

plt.title('Purchase amount Distribution')

print ('Skew is:', dataframe.Purchase.skew())

print('Kurtosis: %f' % dataframe.Purchase.kurt())

#check which features are numeric

numeric\_features = dataframe.select\_dtypes(include=[np.number])

numeric\_features.dtypes

#ds = pd.Series({"Column" : dataframe})

#create plots for each predictor to assess its distribution

sns.set(style="darkgrid")

ax = sns.countplot(dataframe.Occupation)

# ax.set\_xticklabels(ax.get\_xticklabels(), rotation=40, ha="right", fontsize =7)

# kind = “bar”, “strip”, “swarm”, “box”, “violin”, or “boxen”.

sns.countplot(dataframe.Occupation, hue= dataframe.Gender) #hue determine how data is plotted

sns.countplot(dataframe.Marital\_Status)

sns.countplot(dataframe.Gender)

sns.countplot(dataframe.Product\_Category\_1, orient = 'h')

sns.countplot(dataframe.Product\_Category\_2, orient = 'h')

sns.countplot(dataframe.Product\_Category\_3, orient = 'h')

sns.countplot(dataframe.Age)

sns.countplot(dataframe.Stay\_In\_Current\_City\_Years)

sns.countplot(dataframe.City\_Category)

#Correlation between Numerical Predictors and Target variable

corr = numeric\_features.corr()

print (corr['Purchase'].sort\_values(ascending=False)[:10], '\n')

print (corr['Purchase'].sort\_values(ascending=False)[-10:])

f, ax = plt.subplots(figsize=(20, 9))

sns.heatmap(corr, vmax=.8,annot\_kws={'size': 20}, annot=True)

"""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""

Data Preprocessing and Feature Engineering Section

"""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""

#at first, we are checking the percentage of missing value for each cloumns

dataframe.isnull().sum()/dataframe.shape[0]\*100

dataframe = dataframe.drop(['User\_ID','Product\_ID'],axis = 1) #drop the first two columns since they are irrelavant

dataframe.shape

#dataframe['Product\_Category\_2'] == np.nan

#fill NaN with 0 instead

dataframe['Product\_Category\_2']= \

dataframe['Product\_Category\_2'].fillna(0.0).astype("float")

dataframe['Product\_Category\_3']= \

dataframe['Product\_Category\_3'].fillna(0.0).astype("float")

#dataframe = dataframe.replace(np.nan,0)

dataframe.Product\_Category\_2.value\_counts().sort\_index()

dataframe.Product\_Category\_3.value\_counts().sort\_index()

#get the index of all columns equal to 19,20

condition = dataframe.index[(dataframe.Product\_Category\_1.isin([19,20]))]

data = dataframe.drop(condition)

data.Product\_Category\_1.value\_counts().sort\_index()

data.apply(lambda x: len(x.unique()))

#feature engineering

#Turn gender into binary

gender\_dict = {'F':0, 'M':1}

data['Gender'] = data['Gender'].apply(lambda line: gender\_dict[line])

data['Gender'].value\_counts()

#convert age to numeric values

age\_dict = {'0-17':0, '18-25':1, '26-35':2, '36-45':3, '46-50':4, '51-55':5, '55+':6}

data['Age'] = data['Age'].apply(lambda line: age\_dict[line])

#conver city\_category to binary

city\_dict = {'A':0, 'B':1, 'C':2}

data['City\_Category'] = data['City\_Category'].apply(lambda line: city\_dict[line])

#create dummmy variables for Stay\_In\_Current\_City\_Years

le = LabelEncoder()

data['Stay\_In\_Current\_City\_Years'] = le.fit\_transform(data['Stay\_In\_Current\_City\_Years'])

data = pd.get\_dummies(data, columns=['Stay\_In\_Current\_City\_Years'])

data.dtypes

#we are takking 5000 sample after we shuffle the orignal data np array

dataset = data.values

np.random.shuffle(dataset)

dataset = dataset[:5000]

dataset.shape

"""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""

Data Partition Section

"""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""

#(data\_train, data\_test) = model\_selection.train\_test\_split(dataset, train\_size = 0.80, random\_state = 7)

x = dataset[:,[0,1,2,3,4,5,6,7,9,10,11,12,13]]

y = dataset[:,8]

x\_stand = preprocessing.StandardScaler() #standardize

y\_minmax = preprocessing.MinMaxScaler()

x = np.array(x).reshape((len(x),13))

y = np.array(y).reshape((len(y),1))

x = x\_stand.fit\_transform(x)

y = y\_minmax.fit\_transform(y)

#split the data into training and testing

x\_train, x\_test, y\_train, y\_test = model\_selection.train\_test\_split(x,y, train\_size = 0.80, random\_state = 7)

"""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""

Define Model Section

"""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""

#init-mode optimization

start\_time = datetime.datetime.now()

# Function to create model, required for KerasClassifier

def create\_model(init\_mode='uniform'):

# create model

model = Sequential()

model.add(Dense(12, input\_dim=13, activation='relu'))

model.add(Dense(1, activation='sigmoid'))

optimizer = 'adam'

# Compile model

model.compile(loss='mean\_squared\_error', optimizer=optimizer, metrics=['mse'])

return model

# create model

model = KerasRegressor(build\_fn=create\_model, epochs=100, batch\_size=10, verbose=0)

#different type of initializer mode to choose for grid search

init\_mode = ['uniform', 'lecun\_uniform', 'normal', 'zero', 'glorot\_normal', 'glorot\_uniform', 'he\_normal', 'he\_uniform']

#the parameter has to be dictionary for Grid search

param\_grid = dict(init\_mode=init\_mode)

grid = GridSearchCV(estimator=model, param\_grid=param\_grid, n\_jobs=1, verbose=2)

#fit the model onto

grid\_result = grid.fit(x\_test, y\_test)

print("The best loss is: %f using %s" % (grid\_result.best\_score\_, grid\_result.best\_params\_))

means = grid\_result.cv\_results\_['mean\_test\_score']

stds = grid\_result.cv\_results\_['std\_test\_score']

params = grid\_result.cv\_results\_['params']

for mean, stdev, param in zip(means, stds, params):

print("%f (%f) with: %r" % (mean, stdev, param))

stop\_time = datetime.datetime.now()

print ("Time required for training:",stop\_time - start\_time)

start\_time = datetime.datetime.now()

#Nueron optimization

def create\_model(neurons=1):

# create model

model = Sequential()

model.add(Dense(neurons, input\_dim=13, kernel\_initializer='uniform', activation='relu', kernel\_constraint=maxnorm(4)))

model.add(Dense(1))

# Compile model

model.compile(loss='mean\_squared\_error', optimizer= 'Adamax', metrics=['mae'])

return model

# create model

model = KerasRegressor(build\_fn=create\_model, epochs=100, batch\_size=10, verbose=0)

# define the grid search parameters

neurons = [1, 5, 10, 15, 20, 25, 30]

param\_grid = dict(neurons=neurons)

#https://scikit-learn.org/stable/modules/generated/sklearn.model\_selection.GridSearchCV.html

# n\_jobs=-1 means using all processors (don't use).

grid = GridSearchCV(estimator=model, param\_grid=param\_grid, n\_jobs=1, verbose=1)

grid\_result = grid.fit(x\_train, y\_train)

# print out results

print("The best loss is: %f using %s" % (grid\_result.best\_score\_, grid\_result.best\_params\_))

means = grid\_result.cv\_results\_['mean\_test\_score']

stds = grid\_result.cv\_results\_['std\_test\_score']

params = grid\_result.cv\_results\_['params']

for mean, stdev, param in zip(means, stds, params):

print("%f (%f) with: %r" % (mean, stdev, param))

stop\_time = datetime.datetime.now()

print ("Time required for training:",stop\_time - start\_time)

"""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""

Model Selection Section

"""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""

neurons = [1, 5, 10, 15, 20, 25, 30]

def create\_model(neurons =1):

model = Sequential()

model.add(Dense(neurons, input\_dim=13, kernel\_initializer='uniform', activation='linear'))

model.add(Dense(1, activation='sigmoid'))

optimizer = 'adam'

model.compile(loss='mean\_squared\_error', optimizer=optimizer, metrics=['mse','mae'])

return model

collection\_mse = []

collection\_mae = []

for i in neurons:

model = create\_model(neurons =i)

estimator = model.fit(x\_train, y\_train,epochs=100,verbose=2)

score= model.evaluate(x\_test, y\_test, batch\_size =20, verbose =1)

print("\nTest score:", score[0])

print('Test accuracy:', score[1])

pyplot.plot(estimator.history['mean\_squared\_error'])

pyplot.plot(estimator.history['mean\_absolute\_error'])

pyplot.title('Combined statistics with '+str(i)+' neaurons - MSE & MAE')

pyplot.xlabel('Number of epoch')

pyplot.show()