

E-COMMERCE

Problem Statement

- Amazon is an online shopping website that now caters to millions of people everywhere. Over 34,000 consumer reviews for Amazon brand products like Kindle, Fire TV Stick and more are provided.
- The dataset has attributes like brand, categories, primary categories, reviews.title, reviews.text, and the sentiment. Sentiment is a categorical variable with three levels "Positive", "Negative", and "Neutral". For a given unseen data, the sentiment needs to be predicted.
- You are required to predict Sentiment or Satisfaction of a purchase based on multiple features and review text.

 picture

Dataset Snapshot  picture

Project Task: Week 1

Class Imbalance Problem:

1. Perform an EDA on the dataset.
 - a) See what a positive, negative, and neutral review looks like.
 - b) Check the class count for each class. It's a class imbalance problem.

Simple 0 8 1 Python 3 [3.10] | Idle Mode: Command L1, Col 1 01_Starter_Code.ipynb

2. Convert the reviews in Tf-Idf score.

3. Run multinomial Naive Bayes classifier. Everything will be classified as positive because of the class imbalance.

Project Task: Week 2

Tackling Class Imbalance Problem:

1. Oversampling or undersampling can be used to tackle the class imbalance problem.
2. In case of class imbalance criteria, use the following metrics for evaluating model performance: precision, recall, F1-score, AUC-ROC curve. Use F1-Score as the evaluation criteria for this project.
3. Use Tree-based classifiers like Random Forest and XGBoost. Note: Tree-based classifiers work on two ideologies namely, Bagging or Boosting and have fine-tuning parameter which takes care of the imbalanced class.

Project Task: Week 3

Model Selection:

1. Apply multi-class SVM's and neural nets.
2. Use possible ensemble techniques like: XGboost + oversampled_multinomial_NB.

3. Assign a score to the sentence sentiment (engineer a feature called sentiment score). Use this engineered feature in the model and check for improvements. Draw insights on the same.

Simple 0 8 1 Python 3 [3.10] | Idle Mode: Command L1, Col 1 01_Starter_Code.ipynb

Project Task: Week 4

Applying LSTM:

1. Use LSTM for the previous problem (use parameters of LSTM like top-word, embedding-length, Dropout, epochs, number of layers, etc.) Hint: Another variation of LSTM, GRU (Gated Recurrent Units) can be tried as well.
2. Compare the accuracy of neural nets with traditional ML based algorithms.
3. Find the best setting of LSTM (Neural Net) and GRU that can best classify the reviews as positive, negative, and neutral. Hint: Use techniques like Grid Search, Cross-Validation and Random Search

Optional Tasks: Week 4

Topic Modelling:

1. Cluster similar reviews. Note: Some reviews may talk about the device as a gift-option. Other reviews may be about product looks and some may highlight about its battery and performance. Try naming the clusters.
2. Perform Topic Modelling Hint: Use scikit-learn provided Latent Dirichlet Allocation (LDA) and Non-Negative Matrix Factorization (NMF).

[]: %tensorflow_version 2.x

Simple 0 8 1 Python 3 [3.10] | Idle Mode: Command L1, Col 1 01_Starter_Code.ipynb

```
[ ]: %tensorflow_version 2.x
import tensorflow as tf
print(tf.__version__)

Colab only includes TensorFlow 2.x; %tensorflow_version has no effect.
2.12.0

#Importing Essential Libraries

[ ]: import tensorflow as tf
print(tf.__version__)

2.12.0

[ ]: import warnings
warnings.filterwarnings('ignore')
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
import re
import string
import nltk
import seaborn as sns
from sklearn.model_selection import train_test_split
```

Simple 0 8 1 Python 3 [3.10] | Idle Mode: Command ↵ Ln 1, Col 1 01_Starter_Code.ipynb

```
[ ]: import numpy as np

import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
from wordcloud import WordCloud

from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.feature_extraction.text import CountVectorizer, TfidfVectorizer
from sklearn.naive_bayes import BernoulliNB, MultinomialNB
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn import metrics
from sklearn.metrics import roc_auc_score, accuracy_score
from sklearn.pipeline import Pipeline

from bs4 import BeautifulSoup
import re
import nltk
from nltk.corpus import stopwords
from nltk.stem.porter import PorterStemmer
from nltk.stem import SnowballStemmer, WordNetLemmatizer
from nltk import sent_tokenize, word_tokenize, pos_tag

import logging
from gensim.models import word2vec
from gensim.models.keyedvectors import KeyedVectors
```

Simple 0 8 1 Python 3 [3.10] | Idle Mode: Command ↵ Ln 1, Col 1 01_Starter_Code.ipynb

```
[ ]: from keras.preprocessing import sequence
from keras.preprocessing import sequence
from keras.utils import to_categorical
from keras.models import Sequential
from keras.layers import Dense, Dropout, Activation, Lambda
from tensorflow.keras.layers import LSTM, SimpleRNN, GRU
from keras.preprocessing.text import Tokenizer
from collections import defaultdict
from keras.layers.convolutional import Convolution1D
from keras import backend as K
from keras.layers import Embedding

from keras.callbacks import EarlyStopping

print("Setup Complete")
Setup Complete
```

Week 1 Task

Load the data

Simple 0 8 1 Python 3 [3.10] | Idle Mode: Command ↵ Ln 1, Col 1 01_Starter_Code.ipynb

X https://lms.simplilearn.com
Practice Labs | Data Science with Python

jupyterlab jupyter

File Edit View Run Kernel Tabs Settings Help

Amazon ai capstone.ipynb +

[14]: # Keeping only those features that we need for further exploring.
data1 = data[["sentiment","reviews.text"]]

[15]: data1.head()

	sentiment	reviews.text
0	Positive	Purchased on Black FridayPros - Great Price e...
1	Positive	I purchased two Amazon in Echo Plus and two do...
2	Neutral	Just an average Alexa option. Does show a few ...
3	Positive	very good product. Exactly what I wanted, and ...
4	Positive	This is the 3rd one I've purchased. I've bough...

[16]: # Resetting the Index.
data1.index = pd.Series(list(range(data1.shape[0])))

[17]: print('Shape : ',data1.shape)
data1.head()

Shape : (4000, 2)

	sentiment	reviews.text
0	Positive	Purchased on Black FridayPros - Great Price e...

Python 3 [3.10] | Idle

Mode: Command Ln 1, Col 1 Amazon ai capstone.ipynb

X https://lms.simplilearn.com
Practice Labs | Data Science with Python

jupyterlab jupyter

File Edit View Run Kernel Tabs Settings Help

Amazon ai capstone.ipynb +

Setup complete

[nltk_data] Downloading package wordnet to /voc/work/nltk_data...
[nltk_data] Package wordnet is already up-to-date!
[nltk_data] Downloading package stopwords to /voc/work/nltk_data...
[nltk_data] Package stopwords is already up-to-date!

[19]: data1['Processed_Review'] = data1['reviews.text'].apply(preprocess)

[19]: data1.head()

	sentiment	reviews.text	Processed_Review
0	Positive	Purchased on Black FridayPros - Great Price e...	purchase black fridaypros great price even sal...
1	Positive	I purchased two Amazon in Echo Plus and two do...	purchase two amazon echo plus two dot plus fou...
2	Neutral	Just an average Alexa option. Does show a few ...	average alexa option show thing screen still l...
3	Positive	very good product. Exactly what I wanted, and ...	good product exactly want good price
4	Positive	This is the 3rd one I've purchased. I've bough...	rd one purchase buy one niece case compare one...

[21]: data2 = data1[['sentiment','Processed_Review']]

[21]: data2.head()

	sentiment	Processed_Review
0	Positive	purchase black fridaypros great price even sal...

Python 3 [3.10] | Idle

Mode: Command Ln 1, Col 1 Amazon ai capstone.ipynb

https://lms.simplilearn.com
Practice Labs | Data Science with Python

jupyterlab jupyter

File Edit View Run Kernel Tabs Settings Help

Amazon ai capstone.ipynb + Markdown Python 3 [3.10]

1.2 Convert the reviews in Tf-Idf score.

```
[22]: def textPreprocessing(data2):
    #Remove Punctuation Logic
    import string
    removePunctuation = [char for char in data2 if char not in string.punctuation]
    #Join Chars to form sentences
    sentenceWithoutPunctuation = ''.join(removePunctuation)
    words = sentenceWithoutPunctuation.split()
    #StopwordRemoval
    from nltk.corpus import stopwords
    removeStopwords = [word for word in words if word.lower() not in stopwords.words('english')]

    return removeStopwords
```

```
[23]: data2.groupby('sentiment').describe()
```

	Processed_Review			
	count	unique	top	freq
sentiment				
Negative	93	78	first tablet kindle curious update version dis...	3
Neutral	158	145	nothing spectacular item also nothing majorly ...	2
Positive	3749	3372	give grandkids aye christmas love	4

Simple 0 2 Python 3 [3.10] | Idle Mode: Command Ln 1, Col 1 Amazon ai capstone.ipynb

https://lms.simplilearn.com
Practice Labs | Data Science with Python

jupyterlab jupyter

File Edit View Run Kernel Tabs Settings Help

Amazon ai capstone.ipynb + Markdown Python 3 [3.10]

1.4 Tackling Class Imbalance Problem:

Oversampling or undersampling can be used to tackle the class imbalance problem.

```
[39]: from imblearn.over_sampling import RandomOverSampler
ros = RandomOverSampler(random_state=0)
X_res, Y_res = ros.fit_resample(X, Y)
```

```
[40]: from collections import Counter
print(sorted(Counter(Y_res).items()))
[('Negative', 3749), ('Neutral', 3749), ('Positive', 3749)]
```

```
[41]: X_res.shape, Y_res.shape
```

```
[41]: ((11247, 1), (11247,))
```

```
[42]: #Checking out both old & new data
print('Original dataset shape {}'.format(Counter(Y)))
print('Resampled dataset shape {}'.format(Counter(Y_res)))

Original dataset shape Counter({'Positive': 3749, 'Neutral': 158, 'Negative': 93})
Resampled dataset shape Counter({'Positive': 3749, 'Neutral': 3749, 'Negative': 3749})
```

Simple 0 2 Python 3 [3.10] | Idle Mode: Command Ln 1, Col 1 Amazon ai capstone.ipynb

X https://lms.simplilearn.com
Practice Labs | Data Science with Python

jupyterlab jupyter

File Edit View Run Kernel Tabs Settings Help

Amazon ai capstone.ipynb +

Y1=pd.DataFrame(Y_res,columns=['sentiment'])

[45]: #Merging the X & Y output to Final data
Final_data=pd.concat([X1,Y1],axis=1)
Final_data.head()

[45]:

	Processed_Review	sentiment
0	purchase black fridaypros great price even sal...	Positive
1	purchase two amazon echo plus two dot plus fou...	Positive
2	average alexa option show thing screen still l...	Neutral
3	good product exactly want good price	Positive
4	rd one purchase buy one niece case compare one...	Positive

[46]: Final_data.info()

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 11247 entries, 0 to 11246
Data columns (total 2 columns):
 #   Column           Non-Null Count  Dtype  
 ---  --  
 0   Processed_Review  11247 non-null   object 
 1   sentiment        11247 non-null   object 
 dtypes: object(2)
 memory usage: 175.0 KB
```

Simple 0 2 Python 3 [3.10] | Idle Mode: Command End Lab Reset

Ln 1, Col 1 Amazon ai capstone.ipynb

X https://lms.simplilearn.com
Practice Labs | Data Science with Python

jupyterlab jupyter

File Edit View Run Kernel Tabs Settings Help

Amazon ai capstone.ipynb +

Positive 3749
Neutral 3749
Negative 3749
Name: count, dtype: int64

[47]: Text(0.5, 1.0, 'Distribution of Reviews Sentiment')

Distribution of Reviews Sentiment

Sentiment	Count
Positive	3749
Neutral	3749
Negative	3749

[48]: df = Final_data.sample(frac=0.1, random_state=0)
Dropping missing values

Simple 0 2 Python 3 [3.10] | Idle Mode: Command End Lab Reset

Ln 1, Col 1 Amazon ai capstone.ipynb

X https://lms.simplilearn.com
Practice Labs | Data Science with Python

jupyterlab jupyter

File Edit View Run Kernel Tabs Settings Help

Amazon ai capstone.ipynb +

Dropping missing values
df.dropna(inplace=True)

df.head()

[48]:

	Processed_Review	sentiment
8805	buy think would great read book play game howe...	Neutral
9736	good tablet kid lot appts download game	Neutral
125	item work expect great product	Positive
10143	great beginner like child limit use many apps ...	Neutral
10937	buy kindle past time one come defective port b...	Neutral

Train & Test Split Data

[49]: # Splitting data into training set and validation
X_train, X_test, y_train, y_test = train_test_split(df['Processed_Review'], df['sentiment'],
test_size=0.1, random_state=0)

print('Load %d training examples and %d validation examples. \n' %(X_train.shape[0], X_test.shape[0]))
print('Show a review in the training set : \n', X_train.iloc[10])

Simple 0 2 Python 3 [3.10] | Idle Mode: Command ↵ Ln 1, Col 1 Amazon ai capstone.ipynb

X https://lms.simplilearn.com
Practice Labs | Data Science with Python

jupyterlab jupyter

File Edit View Run Kernel Tabs Settings Help

Amazon ai capstone.ipynb +

[50]: def cleanText(raw_text, remove_stopwords=False, stemming=False, split_text=False, \

):
 ...
 Convert a raw review to a cleaned review
 ...
 text = BeautifulSoup(raw_text, 'lxml').get_text() #remove html
 letters_only = re.sub("[^a-zA-Z]", " ", text) # remove non-character
 words = letters_only.lower().split() # convert to lower case

 if remove_stopwords: # remove stopword
 stops = set(stopwords.words("english"))
 words = [w for w in words if not w in stops]

 if stemming==True: # stemming
 stemmer = PorterStemmer()
 stemmer = SnowballStemmer('english')
 words = [stemmer.stem(w) for w in words]

 if split_text==True: # split text
 return (words)

 return(" ".join(words))

[51]: # Preprocess text data in training set and validation set
X_train_cleaned = []
X_test_cleaned = []

Simple 0 2 Python 3 [3.10] | Idle Mode: Command ↵ Ln 1, Col 1 Amazon ai capstone.ipynb

https://lms.simplilearn.com
Practice Labs | Data Science with Python

jupyterlab jupyter

End Lab Reset

File Edit View Run Kernel Tabs Settings Help

Amazon ai capstone.ipynb +

Python 3 [3.10]

```
X_test_cleaned = []

for d in X_train:
    X_train_cleaned.append(cleanText(d))
print('Show a cleaned review in the training set : \n', X_train_cleaned[10])

for d in X_test:
    X_test_cleaned.append(cleanText(d))

Show a cleaned review in the training set :
daughter love easy navigate hard break

CountVectorizer with Multinomial Naive Bayes (Benchmark Model)

• Now as we have cleaned all reviews the next step is converting the reviews into numerical representations for a machine learning algorithm.

• We will use CountVectorizer which implements both tokenization and occurrence counting in a single class provided by the Sklearn library. The output is a sparse matrix representation of the document.

[52]: from sklearn.feature_extraction.text import CountVectorizer

# Fit and transform the training data to a document-term matrix using CountVectorizer
countVect = CountVectorizer()
X_train_countVect = countVect.fit_transform(X_train_cleaned)

# Access feature names directly from vocabulary_
feature_names = countVect.get_feature_names_out()
```

Simple 0 2 Python 3 [3.10] | Idle Mode: Command Ln 1, Col 1 Amazon ai capstone.ipynb

https://lms.simplilearn.com
Practice Labs | Data Science with Python

jupyterlab jupyter

End Lab Reset

File Edit View Run Kernel Tabs Settings Help

Amazon ai capstone.ipynb +

Python 3 [3.10]

```
print("Number of features:", len(feature_names))
print("Show some feature names:\n", feature_names[::1000]) # Displaying feature names every 1000th element

# SVM
from sklearn.linear_model import SGDClassifier

clf = SGDClassifier(loss="hinge", penalty="l2")
clf.fit(X_train_tfidf, y_train)

Number of features: 691
Show some feature names:
['table']
SGDClassifier()

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

[58]: # Have a look at the top 10 features with the smallest and largest coefficients
feature_names = np.array(tfidf.get_feature_names_out()) # or tfidf.vocabulary_.keys() if get_feature_names_out() is not available
sorted_coef_index = clf.coef_[0].argsort()
print('\nTop 10 features with the smallest coefficients :\n{}\n'.format(feature_names[sorted_coef_index[:10]]))
print('Top 10 features with the largest coefficients :\n{}\n'.format(feature_names[sorted_coef_index[:-11:-1]]))

Top 10 features with the smallest coefficients :
['overall' 'play' 'well' 'sometimes' 'easy' 'great' 'love' 'age' 'unit'
 'look']
```

Simple 0 2 Python 3 [3.10] | Idle Mode: Command Ln 1, Col 1 Amazon ai capstone.ipynb

https://lms.simplilearn.com
Practice Labs | Data Science with Python

jupyterlab jupyter

File Edit View Run Kernel Tabs Settings Help

Amazon ai capstone.ipynb +

Python 3 [3.10]

```
[60]: # Evaluating on the validation set
predictions = clf.predict(tfidf.transform(X_test_cleaned))
modelEvaluation(predictions)

Accuracy on validation set: 0.9115

Classification report :
precision    recall   f1-score   support
Negative      0.93    1.00     0.96     39
Neutral       0.84    0.95     0.89     39
Positive      1.00    0.77     0.87     35

accuracy          0.91    113
macro avg        0.92    0.91     0.91     113
weighted avg     0.92    0.91     0.91     113

Confusion Matrix :
[[39  0  0]
 [ 2 37  0]
 [ 1  7 27]]
```

1.6 Use Tree-based classifiers like Random Forest and XGBoost.

Simple 0 2 Python 3 [3.10] | Idle Mode: Command Ln 1, Col 1 Amazon ai capstone.ipynb

https://lms.simplilearn.com
Practice Labs | Data Science with Python

jupyterlab jupyter

File Edit View Run Kernel Tabs Settings Help

Amazon ai capstone.ipynb +

Python 3 [3.10]

```
eval_metric=None, gamma=0, gpu_id=-1, grow_policy='depthwise',
importance_type=None, interaction_constraints='',
learning_rate=0.300000012, max_bin=256, max_cat_to_onehot=4,
max_delta_step=0, max_depth=6, max_leaves=0, min_child_weight=1,
missing=nan, monotone_constraints='()', n_estimators=100,
n_jobs=0, num_parallel_tree=1, objective='multi:softprob',
predictor='auto', random_state=0, reg_alpha=0, ...)
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

```
[63]: # Look at the top 10 features with smallest and the largest coefficients
feature_names = np.array(tfidf.get_feature_names_out()) # or tfidf.vocabulary_.keys() if get_feature_names_out() is not available
# sorted_coef_index = xgb.coef_[0].argsort()
print('\nTop 10 features with smallest coefficients : \n{}\n'.format(feature_names[sorted_coef_index[:10]]))
print('Top 10 features with largest coefficients : \n{}\n'.format(feature_names[sorted_coef_index[:-11:-1]]))

Top 10 features with smallest coefficients :
['overall' 'play' 'well' 'sometimes' 'easy' 'great' 'love' 'age' 'unit'
 'look']

Top 10 features with largest coefficients :
['terrible' 'return' 'minute' 'poor' 'exchange' 'update' 'th' 'never'
 'case' 'dark']

[64]: # Assuming predictions are in numerical format
```

Simple 0 2 Python 3 [3.10] | Idle Mode: Command Ln 1, Col 1 Amazon ai capstone.ipynb

X https://lms.simplilearn.com
Practice Labs | Data Science with Python

jupyterlab jupyter

File Edit View Run Kernel Tabs Settings Help

Amazon ai capstone.ipynb +

Python 3 [3.10]

```
[64]: # Assuming predictions are in numerical format
predictions = xgb.predict(tfidf.transform(X_test_cleaned))

# Convert numerical predictions to the same type as the original labels
predictions = label_encoder.inverse_transform(predictions)

# Now perform evaluation
modelEvaluation(predictions)

Accuracy on validation set: 0.9292

Classification report :
precision    recall   f1-score   support
Negative      0.95     0.97     0.96      39
Neutral       0.98     0.95     0.92      39
Positive      0.94     0.86     0.90      35

accuracy          0.93
macro avg       0.93     0.93     0.93      113
weighted avg    0.93     0.93     0.93      113

Confusion Matrix :
[[38  0  1]
 [ 1 37  1]
 [ 1  1  20]]
```

Simple 0 2 Python 3 [3.10] | Idle Mode: Command Ln 1, Col 1 Amazon ai capstone.ipynb

X https://lms.simplilearn.com
Practice Labs | Data Science with Python

jupyterlab jupyter

File Edit View Run Kernel Tabs Settings Help

Amazon ai capstone.ipynb +

Python 3 [3.10]

```
if word in index2word_set:
    nwords += 1
    featureVec = np.add(featureVec, model.wv[word])
featureVec = np.divide(featureVec, nwords)
return featureVec

def cleanText(text, remove_stopwords=False, split_text=False):
    # Your cleaning code here
    cleaned_text = text # Example: Placeholder for actual cleaning code
    return cleaned_text

X_train_cleaned = []
for review in X_train:
    X_train_cleaned.append(cleanText(review, remove_stopwords=True, split_text=True))
trainVector = getAvgFeatureVecs(X_train_cleaned, w2v, num_features)
print("Training set : %d feature vectors with %d dimensions" % trainVector.shape)

# Getting feature vectors for validation set
X_test_cleaned = []
for review in X_test:
    X_test_cleaned.append(cleanText(review, remove_stopwords=True, split_text=True))
testVector = getAvgFeatureVecs(X_test_cleaned, w2v, num_features)
print("Validation set : %d feature vectors with %d dimensions" % testVector.shape)

Training set : 1012 feature vectors with 300 dimensions
Validation set : 113 feature vectors with 300 dimensions
```

Random Forest Classifier

Simple 0 2 Python 3 [3.10] | Idle Mode: Command Ln 1, Col 1 Amazon ai capstone.ipynb

X https://lms.simplilearn.com
Practice Labs | Data Science with Python

jupyterlab jupyter

File Edit View Run Kernel Tabs Settings Help

Amazon ai capstone.ipynb + Code Python 3 [3.10]

```
for review in X_test:  
    X_test_cleaned.append(cleanText(review, remove_stopwords=True, split_text=True))  
testVector = getAvgFeatureVecs(X_test_cleaned, w2v, num_features)  
print("Validation set : %d feature vectors with %d dimensions" % testVector.shape)  
  
Training set : 1012 feature vectors with 300 dimensions  
Validation set : 113 feature vectors with 300 dimensions
```

Random Forest Classifier

We will now train the Random Forest Classifier using feature vectors of reviews in the training set.

```
[72]: from sklearn.impute import SimpleImputer  
  
# Initialize SimpleImputer  
imputer = SimpleImputer(strategy='mean')  
  
# Impute missing values in trainVector and testVector  
trainVector_imputed = imputer.fit_transform(trainVector)  
testVector_imputed = imputer.transform(testVector)  
  
# Train RandomForestClassifier  
rf = RandomForestClassifier(n_estimators=100)  
rf.fit(trainVector_imputed, y_train)  
predictions = rf.predict(testVector_imputed)  
modelEvaluation(predictions)
```

Simple 0 2 Python 3 [3.10] | Idle Mode: Command Ln 1, Col 1 Amazon ai capstone.ipynb

X https://lms.simplilearn.com
Practice Labs | Data Science with Python

jupyterlab jupyter

File Edit View Run Kernel Tabs Settings Help

Amazon ai capstone.ipynb + Code Python 3 [3.10]

Long Short Term Memory(LSTM) Networks are a special kind of the Recurrent Neural Networks(RNN) capable of learning long-term dependencies. LSTM can be very useful in text mining problems as it involves dependencies in the sentences which can be caught in the "memory" of the LSTM. Here, we will train a simple LSTM and LSTM with Word2Vec embedding for classifying the reviews into positive and negative sentiments using Keras library.

Simple LSTM

We need to preprocess the text data to 2D tensor before we begin fitting it into a simple LSTM. Firstly we will tokenize the corpus by considering only top words (top_words = 20000) and transforming reviews to numerical sequences using the trained tokenizer. Lastly we will make it sure that all the numerical sequences have the same length (maxlen=100) for modelling by truncating the long reviews and padding shorter reviews having zero values.

For constructing a simple LSTM, we will use embedding class in Keras to building up the first layer. This embedding layer converts numerical sequence of words into a word embedding. We should also note that the embedding class provides a convenient way to map discrete words into a continuous vector space but it doesn't take the semantic similarity of the words into account. The next layer is the LSTM layer with 128 memory units. Finally, we will use a dense output layer with a single neuron and a sigmoid activation function to make 0 or 1 prediction for the two classes (positive sentiment and negative sentiment). As it is a binary classification problem log loss is used as the loss function(binary_crossentropy in Keras). ADAM optimization algorithm will be used.

Here's the workflow in this part:-

1. Prepare X_train and X_test to 2D tensor.
2. Train a simple LSTM (embedding layer => LSTM layer => dense layer).
3. Compile and fit the model using log loss function and ADAM optimizer.

```
[73]: 15. Final data format/Final_0_1_random_state_0
```

Simple 0 2 Python 3 [3.10] | Idle Mode: Command Ln 1, Col 1 Amazon ai capstone.ipynb

X https://lms.simplilearn.com
Practice Labs | Data Science with Python

jupyterlab jupyter

File Edit View Run Kernel Tabs Settings Help

Amazon ai capstone.ipynb +

[75]:

```
from tensorflow.keras.preprocessing.sequence import pad_sequences
from tensorflow.keras.utils import to_categorical

top_words = 20000
maxlen = 100
batch_size = 32
nb_classes = 3
nb_epoch = 3

# Vectorize X_train and X_test to 2D tensor
tokenizer = Tokenizer(nb_words=top_words) #Considering only top 20000 words in the corpus
tokenizer.fit_on_texts(X_train)
sequences_train = tokenizer.texts_to_sequences(X_train)
sequences_test = tokenizer.texts_to_sequences(X_test)

X_train_seq = pad_sequences(sequences_train, maxlen=maxlen)
X_test_seq = pad_sequences(sequences_test, maxlen=maxlen)

# One-Hot Encoding of y_train and y_test
y_train_seq = to_categorical(y_train, nb_classes)
y_test_seq = to_categorical(y_test, nb_classes)

print('X_train shape:', X_train_seq.shape) #(27799, 100)
print('X_test shape:', X_test_seq.shape) #(3089, 100)
print('y_train shape:', y_train_seq.shape) #(27799, 3)
print('y_test shape:', y_test_seq.shape) #(3089, 3)

X_train shape: (1012, 100)
X_test shape: (113, 100)
```

Simple 0 2 Python 3 [3.10] | Idle Mode: Command ↻ Ln 1, Col 1 Amazon ai capstone.ipynb

X https://lms.simplilearn.com
Practice Labs | Data Science with Python

jupyterlab jupyter

File Edit View Run Kernel Tabs Settings Help

Amazon ai capstone.ipynb +

[76]:

```
from keras.layers import Embedding, LSTM, Dense, Activation, Dropout
from keras.models import Sequential

# Constructing a Simple LSTM
model1 = Sequential()
model1.add(Embedding(top_words, 128))
model1.add(Dropout(0.2)) # Adding Dropout regularization after Embedding Layer
model1.add(LSTM(128, dropout=0.2, recurrent_dropout=0.2)) # Recurrent dropout
model1.add(Dense(nb_classes))
model1.add(Activation('softmax'))
model1.summary()

# Compiling LSTM
model1.compile(loss='binary_crossentropy',
                optimizer='adam',
                metrics=['accuracy'])

model1.fit(X_train_seq, y_train_seq, batch_size=batch_size, epochs=nb_epoch, verbose=1) # Use epochs instead of nb_epoch

# Model Evaluation
score = model1.evaluate(X_test_seq, y_test_seq, batch_size=batch_size)
print('Test loss : {:.4f}'.format(score[0]))
print('Test accuracy : {:.4f}'.format(score[1]))
```

Simple 0 2 Python 3 [3.10] | Idle Mode: Command ↻ Ln 1, Col 1 Amazon ai capstone.ipynb

X https://lms.simplilearn.com Practice Labs | Data Science with Python

jupyterlab jupyter

File Edit View Run Kernel Tabs Settings Help

Amazon ai capstone.ipynb + Python 3 [3.10]

```
Non-trainable params: 0 (0.00 Byte)

Epoch 1/3
32/32 [=====] - 9s 207ms/step - loss: 0.6434 - accuracy: 0.4051
Epoch 2/3
32/32 [=====] - 7s 207ms/step - loss: 0.5430 - accuracy: 0.6245
Epoch 3/3
32/32 [=====] - 7s 207ms/step - loss: 0.3220 - accuracy: 0.8567
4/4 [=====] - 0s 13ms/step - loss: 0.2705 - accuracy: 0.8496
Test loss : 0.2705
Test accuracy : 0.8496
```

LSTM with Word2Vec Embedding

In the simple LSTM model constructed above, the embedding class in Keras comes in handy for converting the numerical sequence of words into a word embedding but it doesn't take the semantic similarity of the words into account. The model assigns random weights to the embedding layer and learn the embeddings by minimizing the global error of the network.

Instead of using random weights we will use pretrained word embeddings for initializing the weight of an embedding layer. Here, we will use the Word2Vec embedding trained in Part 4 for initializing the weights of embedding layer in LSTM.

1. Load pretrained word embedding model.
2. Construct embedding layer using embedding matrix as weights.
3. Train a LSTM with Word2Vec embedding (embedding layer => LSTM layer => dense layer).
4. Compile and fit the model using log loss function and ADAM optimizer.

Simple 0 2 Python 3 [3.10] | Idle Mode: Command End Lab Reset

X https://lms.simplilearn.com Practice Labs | Data Science with Python

jupyterlab jupyter

File Edit View Run Kernel Tabs Settings Help

Amazon ai capstone.ipynb + Python 3 [3.10]

```
4. Compile and fit the model using log loss function and ADAM optimizer.
```

```
[77]: # Loading pretrained Word2Vec model
w2v = Word2Vec.load("w2v_300features_10minwordcounts_10context")

# Getting Word2Vec embedding matrix
embedding_matrix = w2v.wv.vectors # embedding matrix, type = numpy.ndarray
print("Shape of embedding matrix : ", embedding_matrix.shape) # (vocabular size, embedding dimension)
# w2v.wv.vectors[0] # feature vector of the first word in the vocabulary list

Shape of embedding matrix : (416, 300)

[78]: from tensorflow.keras.utils import to_categorical
```

```
# Your existing code here
top_words = embedding_matrix.shape[0] # 4016
maxlen = 100
batch_size = 32
nb_classes = 3
nb_epoch = 3

# Vectorizing X_train and X_test to 2D tensor
tokenizer = Tokenizer(nb_words=top_words) # Considering only top 20000 words in the corpus
tokenizer.fit_on_texts(X_train)
# tokenizer.word_index # access word-to-index dictionary of trained tokenizer

sequences_train = tokenizer.texts_to_sequences(X_train)
sequences_test = tokenizer.texts_to_sequences(X_test)
```

Simple 0 2 Python 3 [3.10] | Idle Mode: Command End Lab Reset

X https://lms.simplilearn.com
Practice Labs | Data Science with Python

jupyterlab jupyter

File Edit View Run Kernel Tabs Settings Help

Amazon ai capstone.ipynb +

Code

Python 3 [3.10]

```
embedding_matrix.shape[1], # 300
weights=[embedding_matrix])

# Constructing LSTM with Word2Vec embedding
model2 = Sequential()
model2.add(embedding_layer)
model2.add(LSTM(128, dropout=0.2, recurrent_dropout=0.2)) # Fix dropout arguments
model2.add(Dense(nb_classes))
model2.add(Activation('softmax'))
model2.summary()

# Compiling model
model2.compile(loss='binary_crossentropy',
                optimizer='adam',
                metrics=['accuracy'])

model2.fit(X_train_seq, y_train_seq, batch_size=batch_size, epochs=nb_epoch, verbose=1) # Fix 'nb_epoch' to 'epochs'

# Model evaluation
score = model2.evaluate(X_test_seq, y_test_seq, batch_size=batch_size)
print('Test loss : {:.4f}'.format(score[0]))
print('Test accuracy : {:.4f}'.format(score[1]))
Model: "sequential_2"
Layer (type)          Output Shape         Param #
=====
```

Simple 0 2 Python 3 [3.10] | Idle Mode: Command Ln 1, Col 1 Amazon ai capstone.ipynb

X https://lms.simplilearn.com
Practice Labs | Data Science with Python

jupyterlab jupyter

File Edit View Run Kernel Tabs Settings Help

Amazon ai capstone.ipynb +

Code

Python 3 [3.10]

```
[79]: # Constructing Word2Vec embedding layer
embedding_layer = Embedding(embedding_matrix.shape[0], # 4016
                            embedding_matrix.shape[1], # 300
                            weights=[embedding_matrix])

# Constructing LSTM with Word2Vec embedding
model2 = Sequential()
model2.add(embedding_layer)
model2.add(LSTM(128, dropout=0.2, recurrent_dropout=0.2)) # Fix dropout arguments
model2.add(Dense(nb_classes))
model2.add(Activation('softmax'))
model2.summary()

# Compiling model
model2.compile(loss='binary_crossentropy',
                optimizer='adam',
                metrics=['accuracy'])

model2.fit(X_train_seq, y_train_seq, batch_size=batch_size, epochs=nb_epoch, verbose=1) # Fix 'nb_epoch' to 'epochs'

# Model evaluation
score = model2.evaluate(X_test_seq, y_test_seq, batch_size=batch_size)
print('Test loss : {:.4f}'.format(score[0]))
print('Test accuracy : {:.4f}'.format(score[1]))
Model: "sequential_1"
```

Simple 0 2 Python 3 [3.10] | Idle Mode: Command Ln 1, Col 1 Amazon ai capstone.ipynb

X https://lms.simplilearn.com
Practice Labs | Data Science with Python

jupyterlab jupyter

File Edit View Run Kernel Tabs Settings Help

Amazon ai capstone.ipynb +

4/4 [=====] - 0s 23ms/step - loss: 0.2527 - accuracy: 0.8850
Test loss : 0.2527
Test accuracy : 0.8850

[81]: # Getting weight matrix of the embedding Layer
print("Size of weight matrix in the embedding layer : ", \\\nmodel2.layers[0].get_weights()[0].shape) #(20000, 128)

Getting weight matrix of the hidden Layer
print("Size of weight matrix in the hidden layer : ", \\\nmodel2.layers[1].get_weights()[0].shape) #(128, 512) weight dim of LSTM - w

Getting weight matrix of the output Layer
print("Size of weight matrix in the output layer : ", \\\nmodel2.layers[2].get_weights()[0].shape) #(128, 2) weight dim of dense Layer

Size of weight matrix in the embedding layer : (416, 300)
Size of weight matrix in the hidden layer : (300, 512)
Size of weight matrix in the output layer : (128, 3)

Optional Tasks: Topic Modelling

Latent Dirichlet Allocation(LDA)

[82]: import nltk
nltk.download('wordnet')

Simple 0 2 Python 3 [3.10] | Idle Mode: Command L 1, Col 1 Amazon ai capstone.ipynb

X https://lms.simplilearn.com
Practice Labs | Data Science with Python

jupyterlab jupyter

File Edit View Run Kernel Tabs Settings Help

Amazon ai capstone.ipynb +

Latent Dirichlet Allocation(LDA)

[82]: import nltk
nltk.download('wordnet')

doc_complete = data2["Processed_Review"].tolist()
doc_clean = [cleanText(doc).split() for doc in doc_complete]

[nltk_data] Downloading package wordnet to /voc/work/nltk_data...
[nltk_data] Package wordnet is already up-to-date!

[83]: import gensim
from gensim import corpora

[84]: dictionary = corpora.Dictionary(doc_clean)
print(dictionary)

Dictionary:3415 unique tokens: ['able', 'access', 'accomplish', 'ad', 'add']...

[87]: doc_term_matrix = [dictionary.doc2bow(doc) for doc in doc_clean]
doc_term_matrix

[87]: [[(0, 1),
(1, 1),
(2, 1),
(3, 1),
(4, 2)]

Simple 0 2 Python 3 [3.10] | Idle Mode: Command L 1, Col 1 Amazon ai capstone.ipynb

https://lms.simplilearn.com
Practice Labs | Data Science with Python

jupyterlab jupyter

End Lab Reset

File Edit View Run Kernel Tabs Settings Help

Amazon ai capstone.ipynb +

Python 3 [3.10]

```
(15, 1),
(16, 2),
(17, 1),
(18, 1),
(19, 1),
(20, 1),
(21, 1),
(22, 1),
(23, 1),
(24, 1),
(25, 1),
(26, 1),
(27, 1),
(28, 1),
(29, 1),
(30, 1),
(31, 1),
(32, 1),
(33, 2),
(34, 1)],
[(6, 1),
(26, 2),
(34, 1),
(35, 4),
(36, 1),
(37, 2),
(38, 1),
(39, 1)],
```

Simple 0 2 Python 3 [3.10] | Idle Mode: Command Ln 1, Col 1 Amazon ai capstone.ipynb

https://lms.simplilearn.com
Practice Labs | Data Science with Python

jupyterlab jupyter

End Lab Reset

File Edit View Run Kernel Tabs Settings Help

Amazon ai capstone.ipynb +

Python 3 [3.10]

```
(56, 1),
(57, 1),
(58, 1),
(59, 1),
(60, 1),
(61, 1),
(62, 1),
(63, 1),
(64, 1),
(65, 1),
(66, 1),
(67, 1),
(68, 1),
(69, 2),
(70, 1),
(71, 1),
(72, 1),
(73, 1),
(74, 2),
(75, 1),
(76, 1),
(77, 1),
(78, 2),
(79, 1),
(80, 1),
(81, 1),
(82, 1),
(83, 1)],
```

Simple 0 2 Python 3 [3.10] | Idle Mode: Command Ln 1, Col 1 Amazon ai capstone.ipynb

https://lms.simplilearn.com
Practice Labs | Data Science with Python

jupyterlab jupyter

File Edit View Run Kernel Tabs Settings Help

Amazon ai capstone.ipynb +

Python 3 [3.10]

```
(36, 1),  
(95, 1),  
(99, 1),  
(103, 1),  
(110, 1),  
(139, 1),  
(153, 1),  
(160, 1),  
(164, 1),  
(175, 1),  
(176, 1),  
(177, 1),  
(178, 1),  
(179, 1),  
(180, 1),  
(181, 1),  
(182, 1),  
(183, 1)],  
[(7, 1),  
(16, 1),  
(26, 1),  
(93, 1),  
(120, 1),  
(128, 1),  
(133, 2),  
(142, 1),  
(184, 1),  
(185, 1),
```

Simple 0 2 Python 3 [3.10] | Idle Mode: Command Ln 1, Col 1 Amazon ai capstone.ipynb

https://lms.simplilearn.com
Practice Labs | Data Science with Python

jupyterlab jupyter

File Edit View Run Kernel Tabs Settings Help

Amazon ai capstone.ipynb +

Python 3 [3.10]

```
(388, 1),  
(390, 1),  
(391, 1),  
(392, 1),  
(393, 1),  
(394, 1),  
(395, 1),  
(396, 1),  
(397, 1),  
(398, 1),  
(399, 1),  
(400, 1),  
(401, 1),  
(402, 1),  
(403, 1),  
(404, 1)],  
[(19, 1),  
(33, 1),  
(49, 1),  
(77, 1),  
(92, 1),  
(94, 1),  
(139, 1),  
(179, 1),  
(181, 1),  
(192, 1),  
(296, 1),  
(405, 1),  
(406, 1)
```

Simple 0 2 Python 3 [3.10] | Idle Mode: Command Ln 1, Col 1 Amazon ai capstone.ipynb

https://lms.simplilearn.com
Practice Labs | Data Science with Python

jupyterlab jupyter

End Lab Reset

File Edit View Run Kernel Tabs Settings Help

Amazon ai capstone.ipynb +

Code

Python 3 [3.10]

```
(22, 1),  
(34, 1),  
(36, 1),  
(62, 1),  
(63, 1),  
(87, 2),  
(88, 1),  
(183, 1),  
(186, 1),  
(133, 1),  
(164, 1),  
(165, 1),  
(176, 1),  
(200, 1),  
(281, 1),  
(243, 1),  
(247, 1),  
(249, 1),  
(293, 1),  
(300, 1),  
(329, 1),  
(481, 1),  
(432, 1),  
(433, 1),  
(434, 1),  
(435, 1),  
(436, 1),  
(437, 1),
```

Simple 0 2 Python 3 [3.10] | Idle Mode: Command Ln 1, Col 1 Amazon ai capstone.ipynb

https://lms.simplilearn.com
Practice Labs | Data Science with Python

jupyterlab jupyter

End Lab Reset

File Edit View Run Kernel Tabs Settings Help

Amazon ai capstone.ipynb +

Code

Python 3 [3.10]

```
(22, 1),  
(79, 1),  
(183, 1),  
(114, 1),  
(128, 1),  
(152, 1),  
(343, 1),  
(346, 1)],  
[(16, 3),  
(35, 1),  
(36, 1),  
(47, 1),  
(64, 1),  
(65, 1),  
(71, 1),  
(79, 1),  
(88, 1),  
(185, 1),  
(217, 2),  
(234, 1),  
(264, 1),  
(296, 1),  
(325, 1),  
(331, 1),  
(357, 1),  
(396, 1),  
(429, 1),  
(473, 1),  
(474, 2)]
```

Simple 0 2 Python 3 [3.10] | Idle Mode: Command Ln 1, Col 1 Amazon ai capstone.ipynb

https://lms.simplilearn.com
Practice Labs | Data Science with Python

jupyterlab jupyter

End Lab Reset

File Edit View Run Kernel Tabs Settings Help

Amazon ai capstone.ipynb +

Code

Python 3 [3.10]

```
[16, 1),  
(34, 1),  
(55, 1),  
(79, 1),  
(185, 1),  
(197, 2),  
(201, 1),  
(213, 1),  
(263, 2),  
(782, 1),  
(1171, 1)],  
[(4, 1),  
(16, 1),  
(35, 1),  
(47, 1),  
(69, 1),  
(179, 1),  
(197, 1),  
(281, 1),  
(211, 1),  
(262, 1),  
(386, 1),  
(664, 1),  
(1912, 1)],  
[(0, 1),  
(4, 1),  
(6, 1),  
(7, 1),
```

Simple 0 2 Python 3 [3.10] | Idle

Mode: Command Ln 1, Col 1 Amazon ai capstone.ipynb

https://lms.simplilearn.com
Practice Labs | Data Science with Python

jupyterlab jupyter

End Lab Reset

File Edit View Run Kernel Tabs Settings Help

Amazon ai capstone.ipynb +

Code

Python 3 [3.10]

```
[16, 1),  
(34, 1),  
(55, 1),  
(79, 1),  
(185, 1),  
(197, 2),  
(201, 1),  
(213, 1),  
(263, 2),  
(782, 1),  
(1171, 1)],  
[(4, 1),  
(16, 1),  
(35, 1),  
(47, 1),  
(69, 1),  
(179, 1),  
(197, 1),  
(281, 1),  
(211, 1),  
(262, 1),  
(386, 1),  
(664, 1),  
(1912, 1)],  
[(0, 1),  
(4, 1),  
(6, 1),  
(7, 1),
```

Simple 0 2 Python 3 [3.10] | Idle

Mode: Command Ln 1, Col 1 Amazon ai capstone.ipynb

The screenshot shows a Jupyter Notebook interface with the following details:

- Title Bar:** https://lms.simplilearn.com Practice Labs | Data Science with Python
- Header Buttons:** End Lab, Reset, and a user icon.
- File Menu:** File, Edit, View, Run, Kernel, Tabs, Settings, Help.
- Code Cell:** The cell title is "Amazon ai capstone.ipynb". It contains the following Python code:

```
[(16, 1),
 (34, 1),
 (55, 1),
 (79, 1),
 (185, 1),
 (197, 2),
 (201, 1),
 (213, 1),
 (263, 2),
 (702, 1),
 (1171, 1)],
 [(4, 1),
 (16, 1),
 (35, 1),
 (47, 1),
 (69, 1),
 (179, 1),
 (197, 1),
 (201, 1),
 (211, 1),
 (262, 1),
 (305, 1),
 (664, 1),
 (1912, 1)],
 [(0, 1),
 (4, 1),
 (6, 1),
 (7, 1)].
```
- Kernel:** Python 3 [3.10].
- Bottom Status Bar:** Mode: Command, Ln 1, Col 1, Amazon ai capstone.ipynb.

https://lms.simplilearn.com
Practice Labs | Data Science with Python

jupyterlab ● jupyter ● End Lab Reset

File Edit View Run Kernel Tabs Settings Help

Amazon ai capstone.ipynb + Code Python 3 [3.10]

```
[91]: word_dict = {}
for i in range(NUM_TOPICS):
    words = ldaModel.show_topic(i, topn = 20)
    word_dict["Topic # " + "{}".format(i)] = [i[0] for i in words]

[92]: pd.DataFrame(word_dict)
```

	Topic # 0	Topic # 1	Topic # 2	Topic # 3	Topic # 4	Topic # 5	Topic # 6	Topic # 7	Topic # 8
0	echo	great	love	easy	tablet	kindle	good	battery	screen
1	alexa	use	buy	love	read	one	sound	charge	amazon
2	show	product	tablet	use	need	buy	great	life	good
3	music	work	old	read	great	get	quality	use	tablet
4	great	easy	year	size	kindle	fire	speaker	read	store
5	love	recommend	kid	purchase	price	amazon	like	tablet	device
6	home	tablet	game	much	good	love	tap	long	apps
7	use	good	play	kindle	book	use	portable	kindle	like
8	device	price	get	book	game	time	price	would	google
9	amazon	would	gift	screen	light	new	product	good	use

Simple Mode: Command Python 3 [3.10] | Idle Ln 1, Col 1 Amazon ai capstone.ipynb

https://lms.simplilearn.com
Practice Labs | Data Science with Python

jupyterlab jupyter

End Lab Reset

File Edit View Run Kernel Tabs Settings Help

Amazon ai capstone.ipynb + Code Python 3 [3.10]

[95]: pip install --upgrade pip

Defaulting to user installation because normal site-packages is not writeable
Requirement already satisfied: pip in ./local/lib/python3.10/site-packages (24.3.1)
Note: you may need to restart the kernel to use updated packages.

Displaying Results & Getting Insights

[96]: import pyLDAvis.gensim

[97]: # Import PyLDAvis
import pyLDAvis.gensim

Set workers parameter to 1 to avoid parallel execution
Lda_display = pyLDAvis.gensim.prepare(ldamodel, doc_term_matrix, dictionary, sort_topics=False, n_jobs=1)

Display the visualization
pyLDAvis.display(Lda_display)

[97]:

Creating a Wordcloud

[]: from wordcloud import WordCloud, STOPWORDS
txt = data2["Processed_Review"].values

Mode: Command Ln 1. Col 1 Amazon ai capstone.ipynb

X https://lms.simplilearn.com
Practice Labs | Data Science with Python

jupyterlab jupyter

End Lab Reset

File Edit View Run Kernel Tabs Settings Help

Amazon ai capstone.ipynb + Code Python 3 [3.10]

```
Non-trainable params: 0 (0.00 Byte)

Epoch 1/3
32/32 [=====] - 9s 207ms/step - loss: 0.6434 - accuracy: 0.4051
Epoch 2/3
32/32 [=====] - 7s 207ms/step - loss: 0.5430 - accuracy: 0.6245
Epoch 3/3
32/32 [=====] - 7s 207ms/step - loss: 0.3220 - accuracy: 0.8567
4/4 [=====] - 0s 13ms/step - loss: 0.2705 - accuracy: 0.8496
Test loss : 0.2705
Test accuracy : 0.8496

LSTM with Word2Vec Embedding

In the simple LSTM model constructed above, the embedding class in Keras comes in handy for converting the numerical sequence of words into a word embedding but it doesn't take the semantic similarity of the words into account. The model assigns random weights to the embedding layer and learn the embeddings by minimizing the global error of the network.

Instead of using random weights we will use pretrained word embeddings for initializing the weight of an embedding layer. Here, we will use the Word2Vec embedding trained in Part 4 for initializing the weights of embedding layer in LSTM.

1. Load pretrained word embedding model.
2. Construct embedding layer using embedding matrix as weights.
3. Train a LSTM with Word2Vec embedding (embedding layer => LSTM layer => dense layer).
4. Compile and fit the model using log loss function and ADAM optimizer.
```

Simple 0 2 Python 3 [3.10] | Idle Mode: Command Ln 1, Col 1 Amazon ai capstone.ipynb