Assessment 2

1) Describe how you would have used the Naïve Bayes algorithm for Project 2 (instead of logistic regression).

Naïve Bayes is a supervised learning algorithm which uses bayes theorem to perform classification operations.
In the project, we have used Logistic regression for performing sentiment analysis by providing 12 features as input features and finding the accuracy by working on training, validation, and testing sets.
The data has corpus with positive and negative scores, adjectives and tweets. We should be able to use all these data for training and predicting with the naïve bayes algorithm.

Steps:
Clean the data : First the data needs to be cleaned. For that the data needs to be preprocessed and remove special characters, emojis, and other types of irrelevant data. All the data will be converted to lower case.
Vocabulary : From the data files read the data and create unique vocabulary with positive and negative values. For better results we can use methods such as TF-IDF.
Dataset : The dataset should be divided into training, validation and test sets. Training set is used to train the naïve bayed classifier where as the test set is used to test the model.
Feature Extraction : for every tweet in the training set, extract the features by using the vocabulary.
Train the Model : By using the extracted features, use them on the training model for training the model.
Test the model : By passing the test data set as the input test the efficiency of the model. We can calculate the accuracy, precision, recall and F1 score and findout how well the model is working.

---

2) In project 2 you used 12 features per tweet and developed a binary classifier. How would you use the same set of features and apply it to a Twitter dataset that has movie ratings with 5 different classes ("Excellent", "Very Good", "Not Bad", "Terrible", "Don't Watch"). Describe the approach you would take to modify your current notebook.

If we want to use the 5 classes "Excellent", "Very Good", "Not Bad", "Terrible", "Don't Watch"…
Data pre-processing
First convert the data by changing the movie rating into numbers. Example
Excellent → 5
Very Good → 4
Not Bad → 3
Terrible → 2
Don't watch → 1
With this the dataset that we pass as input is also changed.
Model Changes

We must change and update the model to handle the new situation where the 5 classes are introduced newly.
The model should be used in a way that the output should classify in five types as ratings like 1 or 2 or 3 or 4 or 5.
Extract Features :
As we have already done in in the project in the same way extract the features for passing as input and along with the newly added.
Model Training :
In the same way, we will use the binary classifier, to train the model and predict the movie rating based on the other 12 features. The output should be modified for the model. It should provide output as 1 or 2 or 3 or 4 or 5.
Test the Model :
Pass the testing data set to the model to find out the performance of the model that is developed.
find the Accuracy, f1 score for evaluating the model.

---

3) Give a pair of examples of similar words and a pair of examples of related words.
Similar Words
Type 1
Compliment – complement
Capitol – capital
Stationery – Stationary
Advice – Advise
Type 2
Big – large
Talk – speak

Related Words
Type 1
Democracy – Democratization
Genetics – genomics
Sociology – socialization
Type 2
Make a difference.
Fish and chips
Fast food

---

4) What is an embedding? Did you use an embedding in Project 2?
It is a technique in NLP where each separate word is considered as a real_valued vector. These are in lower_dimensional_space. This captures the inter_word semantics. Every word in the vector can represent 100's of dimensions. This technique is mostly used in NLP.
In project 2 I have not used embedding.

---

5) How would you construct a term-context matrix. How would you use it?

A term-context matrix, like window of words surrounds the target word, it represents text data that captures the co-occurrence of words inside a specific context. Natural language processing tasks like text categorization and word embedding generation use this kind of model frequently.
Steps:
Text should be tokenized and divided into a list of distinct words.
Find the size of the context window, or the number of words to take into account as the target word's context.
Create a emptymatrix: Create a blank matrix with dimensions m x m,
m = total number of unique words in the text.
loop through the text: For every word in text, consider the words in defined context window. For example, if the context window size is 2, consider the two words before and after the target word.
Update the matrix: every target word and context word pair, increase entry in the term-context matrix by 1.
Normalize matrix : Divide every term in the term-context matrix by adding rows or columns to change the counts into probabilities or relative frequencies.

---

6) Suppose you have training and test data X and Y, where X contains 3 features and Y contains the corresponding labels which are either 0 or 1. Y is the dependent variable. Write pseudo code to train the model and predict use logistic regression.

Answer
Import the required libraries
For logistic regression we need to import LogisticRegression from sklearn.
For holding the data as a data frame use numpy or pandas.
Split the data to training and testing sets
Create an instance of the LogisticRegression. Fit the model.
Train the model.
Test the model.
Find the accuracy of model.
X → has 3 features → independent
Y → has 1 feature → dependent variable
Pseudo Code(python style):

```
# import required Libraries
Import numpy
Import pandas
Import LogisticRegression (from sklearn)
Import train_test_split (from sklearn)
# Split the data
X_train, X_test, Y_train, Y_test = train_test_split(X,Y,test_size, random_state)
# create the model instance
Lr = LogisticRegression()
# train/fit the model
Lr.fit(X_train, Y_train)
```

```
# prediction of model.
Y_pred = Lr.predict(X_test)
# finding accuracy of model. From sklearn
A = accuracy_score(Y_test, Y_pred)
```

---

7) The word “good” appears in all 37 Shakespeare plays. Suppose it appears 170 times in the play “As good as it gets”. What is the tf-idf score of “good” and “As good as it gets”?

Tf → term frequency

Tf(good) in play “As good as it gets” → 170

Idf → inverse document frequency

Idf gives how common the word is.

Idf(good) = Log(N/appearance)

N = 37

Appearance = 37 → appears in all plays.

Log(37/37) = log(1) → 0

Idf(good) = 0

Tf_idf(good) = 170*0 = 0

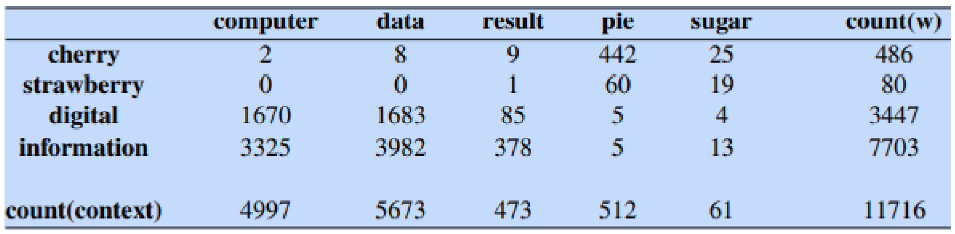
Tf(“As good as it gets”) = 1

Idf(“As good as it gets”) = log(37/1) = 1.5682

Tf_idf(“As good as it gets”) = 1 * 1.5682 → 1.5682

---

8) compute PPMI(digital, result) using data in below table



	computer	data	result	pie	sugar	count(w)
cherry	2	8	9	442	25	486
strawberry	0	0	1	60	19	80
digital	1670	1683	85	5	4	3447
information	3325	3982	378	5	13	7703
count(context)	4997	5673	473	512	61	11716

In general,
PPMI(W, C) = max(0, log2(P(W, C) / (P(W) * P(C))))

Considering the above table as a matrix.

In this Case the below will be the case

PPMI(digital, result) = max(0, log2(P(digital, result) / (P(digital) * P(result))))

When there are 2 words,
PPMI(W1, W2) = max(0, log2(P(W1, W2) / (P(W1) * P(W2))))

Total_word_count = 11716

Digital(count) = 3447

Result(count) = 473

Digital_result(count) = 85

Find probabilities of digital, result → (digital and result)

P(digital) = 3447 / 11716 = 0.2942

P(result) = 473 / 11716 = 0.0403

P(digital,result) = 85 / 11716 = 0.0072

PPMI(digital, result) = max(0, log2(P(digital, result) / (P(digital) * P(result))))

max(0, log2(0.0072 / (0.2942 * 0.0403)))

max(0, log2(0.0072 / 0.0118))

max(0, log2(0.61))

max(0 , -0.7131)

➔ 0

The PPMI score between digital & result is 0.

9) True/False?

a) Cosine similarity is a probability. **False**

b) TF/IDF is used to produce a dense embedding. **False**

c) The output of 'Word2Vec' is a set of word predictions. **False**

d) The 'word2vec' architecture uses a deep (multiple hidden layers) neural network. **True**

10) Compute the sigmoid of $w \cdot x + b$, where the input features are [0.7,0.4,0.5], weights are [0.2,0.3,0.7] and the bias is 0.3.

x = 0.7, 0.4, 0.5 (Inputs)
w = 0.2, 0.3, 0.7 (weights)
b = 0.3 (bias)
σ(y) = 1/ (1+ e^(-y))
Generally, y = w . x + b

Here $y = W_i * X_i + b$

i – i^{th} value

$y = (0.7*0.2 + 0.4*0.3 + 0.5*0.7) + 0.3$

$y = 0.14 + 0.12 + 0.35 + 0.3$

$y = 0.91$

$\sigma(0.91) = 1/(1 + e^{(-0.91)}) = 0.7136$