Creating an Large Language Model (LLM) for Supervised Learning

Description of Data

```
- Review_ID: Reviewer code for dataset.

- Rating: Customer satisfaction rating for the park.

- Year_Month: Date of the review.

- Reviwer_Location: Where the reviewer is from.

- Review_Text: Customer review.

- Branch: Which Dsineyland branch the reviewer is refering to.
```

Objective

Given the reviews for Disneyland Hong Kong and Disneland California, we want to use the reviews of various guests to predict the location. The data set does contain reviews from Disneyland Paris, but we will focus on the two locations listed by restricting the data to the first 25000 reviews within the dataset. This was chosen as the file already had the known locations in order: Disneyland Hong Kong, Disneyland California, and Disneyland Paris respectively. Thus by restricting the data to the first 25000 observations, we will eliminate the Disnleyland Paris entries and reduce the dataset for slighlty faster perfomance of the LLM model.

In [1]:

```
#Packages
import pandas as pd
import numpy as np

# Load the dataset into a DataFrame
disney = pd.read_csv(
    filepath_or_buffer = 'C:\\Users\\BR1NK\\Desktop\\UCF Notes\\STA 5703\\Datasets\
    header = None,
    names=['Review_ID','Rating','Year_Month','Reviewer_Location','Review_Text','Bra
    dtype = str,
    encoding_errors = 'ignore'
)

#Remove the original column names
disney = disney.iloc[1:, :]
#Replace missing values with NaN
disney.replace('missing', np.nan, inplace=True)
#Remove NaN values
```

```
disney = disney.dropna()
#Restrict the data to the first 25000 observations
disney = disney[:25000]
#View the data
disney
```

Out[1]:

	Review_ID	Rating	Year_Month	Reviewer_Location	Review_Text	Brar
1	670772142	4	2019-4	Australia	If you've ever been to Disneyland anywhere you...	Disneyland_HongKc
2	670682799	4	2019-5	Philippines	Its been a while since d last time we visit HK...	Disneyland_HongKc
3	670623270	4	2019-4	United Arab Emirates	Thanks God it wasn t too hot or too humid wh...	Disneyland_HongKc
4	670607911	4	2019-4	Australia	HK Disneyland is a great compact park. Unfortu...	Disneyland_HongKc
5	670607296	4	2019-4	United Kingdom	the location is not in the city, took around 1...	Disneyland_HongKc
...	...	...	...	...	...	
25820	140911281	3	2012-9	United Kingdom	Three E Ticket attractions closed on the same ...	Disneyland_Califor
25821	140892689	5	2012-7	United States	Always a classic time and a place to bring bac...	Disneyland_Califor
25822	140890400	5	2011-10	United States	Have any empty day or weekend and want to do s...	Disneyland_Califor
25823	140876904	4	2012-9	United Kingdom	A great fun place but I think Florida Disney i...	Disneyland_Califor
25824	140873494	5	2012-9	United States	Disneyland is such a magical	Disneyland_Califor

	Review_ID	Rating	Year_Month	Reviewer_Location	Review_Text	Brar
					place that you MU...	

25000 rows × 6 columns

In [2]:
```
#Remove leading and trailing whitespace
disney['Review_Text'] = disney['Review_Text'].str.strip()
#Reset the Index
disney.reset_index(drop=True, inplace=True)
disney.head()
```

Out[2]:

	Review_ID	Rating	Year_Month	Reviewer_Location	Review_Text	Branch
0	670772142	4	2019-4	Australia	If you've ever been to Disneyland anywhere you...	Disneyland_HongKong
1	670682799	4	2019-5	Philippines	Its been a while since d last time we visit HK...	Disneyland_HongKong
2	670623270	4	2019-4	United Arab Emirates	Thanks God it wasn t too hot or too humid wh...	Disneyland_HongKong
3	670607911	4	2019-4	Australia	HK Disneyland is a great compact park. Unfortu...	Disneyland_HongKong
4	670607296	4	2019-4	United Kingdom	the location is not in the city, took around 1...	Disneyland_HongKong

In [3]:
```
#Create numerical labels for the Disneyland Branches
disney['label'] = disney['Branch'].map({'Disneyland_HongKong': 0, 'Disneyland_Calif
disney = disney[['label','Review_Text']]
```

In [4]:
```
#Partition the Data
list_partition = ['Train','Validation','Test']
disney['partition'] = np.random.choice(
    a = list_partition,
    size = disney.shape[0]
)
```

```
#Subgroup the data for easier recall later
X = disney['Review_Text']
y = disney['label']
X_Train = X.loc[disney['partition'] == 'Train']
y_Train = y.loc[disney['partition'] == 'Train']
disney.groupby('partition')['label'].describe()
```

```
C:\Users\BR1NK\AppData\Local\Temp\ipykernel_6132\47929537.py:3: SettingWithCopyWarni
ng:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/u
ser_guide/indexing.html#returning-a-view-versus-a-copy
  disney['partition'] = np.random.choice(
```

Out[4]:

	count	mean	std	min	25%	50%	75%	max
partition								
Test	8216.0	0.635102	0.481431	0.0	0.0	1.0	1.0	1.0
Train	8379.0	0.635875	0.481213	0.0	0.0	1.0	1.0	1.0
Validation	8405.0	0.631410	0.482451	0.0	0.0	1.0	1.0	1.0

In [5]:

```
#Convert the data for the LLM and Logistic Regression
from sklearn.feature_extraction.text import TfidfVectorizer

TfidfVectorizer_disney = TfidfVectorizer().fit(
    raw_documents = X_Train
)
TfidfVectorizer_Train = TfidfVectorizer_disney.transform(
    raw_documents = X_Train
)
TfidfVectorizer_X = TfidfVectorizer_disney.transform(
    raw_documents = X
)
```

In [6]:

```
#Apply Logistic Regression using predicted probabilties of the labels.
from sklearn.linear_model import LogisticRegression
LogisticRegression_disney = LogisticRegression().fit(
    X = TfidfVectorizer_Train,
    y = y_Train
)
LogisticRegression_predict_proba = LogisticRegression_disney.predict_proba(
    X = TfidfVectorizer_X
)
```

In [7]:

```
#Add a column for the prediction probabilities
disney['probability_LogisticRegression'] = pd.DataFrame(LogisticRegression_predict_
C:\Users\BR1NK\AppData\Local\Temp\ipykernel_6132\2098894359.py:1: SettingWithCopyWar
ning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/u
ser_guide/indexing.html#returning-a-view-versus-a-copy
  disney['probability_LogisticRegression'] = pd.DataFrame(LogisticRegression_predict
_proba)[1]
```

In [8]:

```
from datasets import Dataset

# Convert DataFrame to Hugging Face Dataset
Dataset_disney = Dataset.from_pandas(
    df = disney[['Review_Text','label',]]
)
Dataset_Train = Dataset.from_pandas(
    df = disney.loc[disney['partition'] == 'Train',['Review_Text','label',]]
)
Dataset_Validation = Dataset.from_pandas(
    df = disney.loc[disney['partition'] == 'Validation',['Review_Text','label',]]
)
Dataset_Test = Dataset.from_pandas(
    df = disney.loc[disney['partition'] == 'Test',['Review_Text','label',]]
)
```

In [9]:

```
from transformers import AutoTokenizer
# Load pre-trained tokenizer and model
#model_name = "distilbert-base-uncased"
tokenizer = AutoTokenizer.from_pretrained(
    pretrained_model_name_or_path = "distilbert-base-uncased"
)
```

```
C:\Users\BR1NK\anaconda3\Lib\site-packages\transformers\tokenization_utils_base.py:1
601: FutureWarning: `clean_up_tokenization_spaces` was not set. It will be set to `T
rue` by default. This behavior will be depracted in transformers v4.45, and will be
then set to `False` by default. For more details check this issue: https://github.co
m/huggingface/transformers/issues/31884
  warnings.warn(
```

In [10]:

```
#Load pre-trained model for Hugging Face
from transformers import AutoModelForSequenceClassification, Trainer, TrainingArgum
model = AutoModelForSequenceClassification.from_pretrained(
    pretrained_model_name_or_path = "distilbert-base-uncased",
    num_labels = 3
)
```

```
Some weights of DistilBertForSequenceClassification were not initialized from the mo
del checkpoint at distilbert-base-uncased and are newly initialized: ['classifier.bi
as', 'classifier.weight', 'pre_classifier.bias', 'pre_classifier.weight']
You should probably TRAIN this model on a down-stream task to be able to use it for
predictions and inference.
```

In [11]:

```
tokenizer
```

Out[11]:

```
DistilBertTokenizerFast(name_or_path='distilbert-base-uncased', vocab_size=30522,
model_max_length=512, is_fast=True, padding_side='right', truncation_side='right',
special_tokens={'unk_token': '[UNK]', 'sep_token': '[SEP]', 'pad_token': '[PAD]',
'cls_token': '[CLS]', 'mask_token': '[MASK]'}, clean_up_tokenization_spaces=True),
added_tokens_decoder={
        0: AddedToken("[PAD]", rstrip=False, lstrip=False, single_word=False, norm
alized=False, special=True),
        100: AddedToken("[UNK]", rstrip=False, lstrip=False, single_word=False, no
rmalized=False, special=True),
        101: AddedToken("[CLS]", rstrip=False, lstrip=False, single_word=False, no
rmalized=False, special=True),
        102: AddedToken("[SEP]", rstrip=False, lstrip=False, single_word=False, no
rmalized=False, special=True),
        103: AddedToken("[MASK]", rstrip=False, lstrip=False, single_word=False, n
ormalized=False, special=True),
}
```

In [12]:

```python
#Apply tokenizer to each partition
def tokenize_function(examples):
    return tokenizer(
        examples['Review_Text'],
        padding="max_length",
        truncation=True
    )
tokenizer_df = Dataset_disney.map(
    function=tokenize_function,
    batched=True
)
tokenizer_Train = Dataset_Train.map(
    function=tokenize_function,
    batched=True
)
tokenizer_Validation = Dataset_Validation.map(
    function=tokenize_function,
    batched=True
)
tokenizer_Test = Dataset_Test.map(
    function=tokenize_function,
    batched=True
)
```

```
Map:   0%|          | 0/25000 [00:00<?, ? examples/s]
Map:   0%|          | 0/8379 [00:00<?, ? examples/s]
Map:   0%|          | 0/8405 [00:00<?, ? examples/s]
Map:   0%|          | 0/8216 [00:00<?, ? examples/s]
```

In [13]:

```python
#Train the specific model
from transformers import Trainer, TrainingArguments
# Define Trainer
Trainer_Train = Trainer(
    model=model,
    train_dataset=tokenizer_Train,
    eval_dataset=tokenizer_Validation,
).train()
#Save the model
```

```
import pickle
with open('Trainer_model.pkl', 'wb') as file: pickle.dump(Trainer_Train, file)
```

[3144/3144 3:45:14, Epoch 3/3]

Step	Training Loss
500	0.369000
1000	0.272500
1500	0.198200
2000	0.172800
2500	0.105400
3000	0.072200

In [14]:

```
#Open the model (this is if we ever have to rerun the code, we don't need to retrai
import pickle
with open('Trainer_model.pkl', 'rb') as file:
    Trainer_Train = pickle.load(file)
```

In [15]:

```
import torch
DataLoader_df = torch.utils.data.DataLoader(tokenizer_df)
DataLoader_Train = torch.utils.data.DataLoader(tokenizer_Train)
DataLoader_Validation = torch.utils.data.DataLoader(tokenizer_Validation)
DataLoader_Test = torch.utils.data.DataLoader(tokenizer_Test)
```

In [56]:

```
from torch.nn.functional import softmax
model.eval()
list_input_ids = [torch.stack(batch['input_ids']).to(model.device) for batch in Dat
```

In [17]:

```
list_attention_mask = [torch.stack(batch['attention_mask']).to(model.device) for ba
```

In [18]:

```
disney.head()
```

Out[18]:

	label	Review_Text	partition	probability_LogisticRegression
0	0	If you've ever been to Disneyland anywhere you...	Validation	0.169214
1	0	Its been a while since d last time we visit HK...	Validation	0.077608
2	0	Thanks God it wasn t too hot or too humid wh...	Validation	0.161224
3	0	HK Disneyland is a great compact park. Unfortu...	Validation	0.377191
4	0	the location is not in the city, took around 1...	Validation	0.031518

In [19]:
```
#Establishing Prediction probabities
import torch.nn.functional as F
disney['probability_Trainer'] = -1.0
for j in range(disney.shape[0]):
    if disney.loc[j,'probability_Trainer'] < 0:
        disney.loc[j,'probability_Trainer'] = F.softmax(model(
            list_input_ids[j],
            attention_mask = list_attention_mask[j]
        ).logits, dim=1).mean(dim=0)[1].item()
        disney.to_csv('llm_for_classification_supervised_learning.csv')
```

```
C:\Users\BR1NK\AppData\Local\Temp\ipykernel_6132\3252910489.py:2: SettingWithCopyWar
ning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/u
ser_guide/indexing.html#returning-a-view-versus-a-copy
  disney['probability_Trainer'] = -1.0
```

In [20]:
```
df = pd.read_csv('llm_for_classification_supervised_learning.csv')
```

In [22]:
```
#Quick Descrtiption of the data.
df.describe()
```

Out[22]:

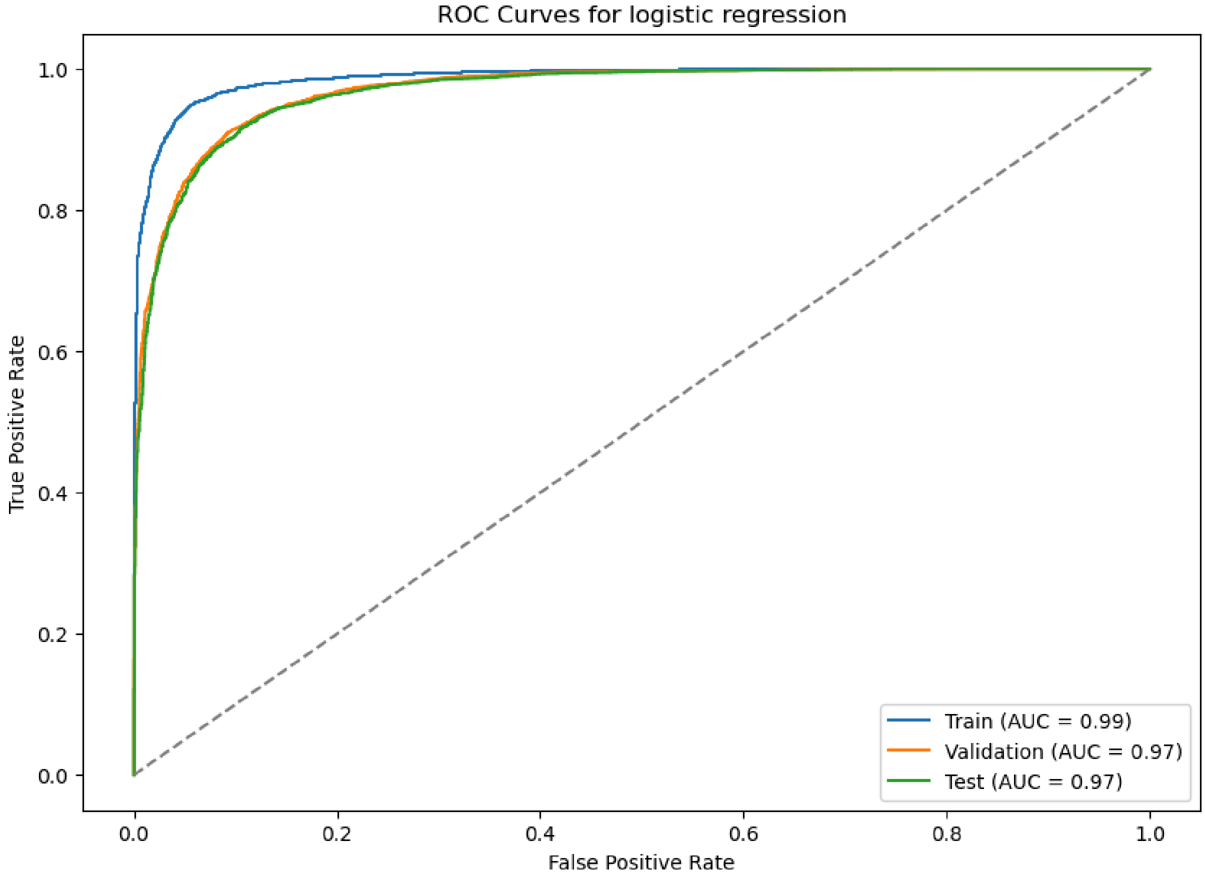
	Unnamed: 0	label	probability_LogisticRegression	probability_Trainer
count	25000.000000	25000.000000	25000.000000	25000.000000
mean	12499.500000	0.634120	0.637941	0.149087
std	7217.022701	0.481686	0.315710	0.042820
min	0.000000	0.000000	0.000303	0.100282
25%	6249.750000	0.000000	0.360360	0.121536
50%	12499.500000	1.000000	0.773467	0.134250
75%	18749.250000	1.000000	0.905976	0.159818
max	24999.000000	1.000000	0.998402	0.387613

In [23]:
```
#Logistic Regression Evaluation
from sklearn.metrics import roc_auc_score
[roc_auc_score(
    y_true = df.loc[df['partition'] == partition,'label'],
    y_score = disney.loc[disney['partition'] == partition,'probability_LogisticRegr
) for partition in list_partition]
```

Out[23]:
```
[0.9871259533870155, 0.9701673599906966, 0.96778937331672]
```

In [24]:
```
#ROC curve of the Logistic Regression Model
import matplotlib.pyplot as plt
from sklearn.metrics import roc_curve, roc_auc_score
```

```
# Initialize plot
plt.figure(figsize=(10, 7))

# Loop through partitions and plot ROC curves
for i, partition in enumerate(list_partition):
    y_true = df.loc[df['partition'] == partition, 'label']
    y_score = disney.loc[disney['partition'] == partition, 'probability_LogisticReg

    fpr, tpr, _ = roc_curve(y_true, y_score)
    auc_score = roc_auc_score(y_true, y_score)

    plt.plot(fpr, tpr, label=f'{list_partition[i]} (AUC = {auc_score:.2f})')

# Plot random chance line
plt.plot([0, 1], [0, 1], color='gray', linestyle='--')

# Add titles and labels
plt.title('ROC Curves for logistic regression')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.legend(loc='lower right')

# Show plot
plt.show()
```



In [25]:

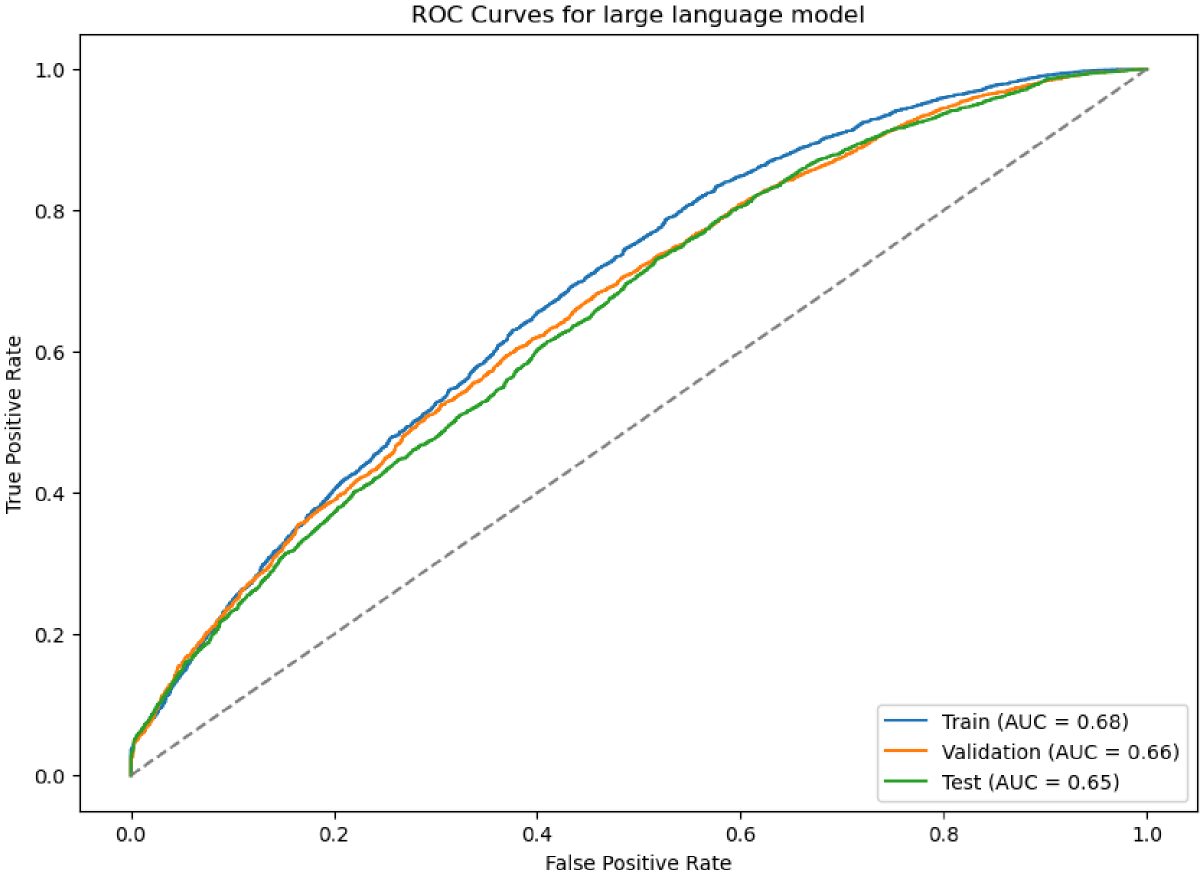
```
#LLM Model Evaluation
[roc_auc_score(
    y_true = df.loc[df['partition'] == partition,'label'],
    y_score = df.loc[df['partition'] == partition,'probability_Trainer']
) for partition in list_partition]
```

Out[25]:

```
[0.6827848620498571, 0.6634746025901208, 0.6533495819750537]
```

In [26]:

```
#ROC curve for LLM
import matplotlib.pyplot as plt
from sklearn.metrics import roc_curve, roc_auc_score

# Initialize plot
plt.figure(figsize=(10, 7))

# Loop through partitions and plot ROC curves
for i, partition in enumerate(list_partition):
    y_true = df.loc[df['partition'] == partition, 'label']
    y_score = df.loc[df['partition'] == partition, 'probability_Trainer']

    fpr, tpr, _ = roc_curve(y_true, y_score)
    auc_score = roc_auc_score(y_true, y_score)

    plt.plot(fpr, tpr, label=f'{list_partition[i]} (AUC = {auc_score:.2f})')

# Plot random chance line
plt.plot([0, 1], [0, 1], color='gray', linestyle='--')

# Add titles and labels
plt.title('ROC Curves for large language model')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.legend(loc='lower right')

# Show plot
plt.show()
```



Thus, it appears that the Logistic Regression Model offered a better prediction for the location of th Disneyland Branch with as signifigantly better AUC score.