Airways Analysis

Project Title: Global Airways Analytics and Predictive Insights

Website: https://www.airlinequality.com/review-pages/a-z-airline-reviews/

Statement

This project aims to use the airline review data to create a visual analytics and develop predictive models. The visualization will enable company to understand airline performance metrics and trends, while the model will predict customer satisfaction and sentiments, and thus to get potential business outcome or early waring by using machine learning and natural language processing techniques.

Research Questions

1. What are the key performance indicators or factors for airlines based on the review?
2. Can the satisfaction be predicted?
3. How do airline company leverage insights from data to improve service and operational efficiency.
4. What value can review content brings if the satisfaction and rating already there?

Process Steps:

- [ ] Part I: Data Collection and Preprocessing
    - [ ] Web Scraping
    - [ ] Cleaning and Preparation
    - [ ] Data Storage
- [ ] Part II: Exploratory Data Analysis
    - [ ] Visualization

- [ ] Word Cloud
- [ ] Part III: Predictive Modeling Comparison
  - [ ] Traditional NLP with Machine Learning
  - [ ] Simple RNN
  - [ ] LSTM
  - [ ] Bidirectional RNN
  - [ ] Transformer
  - [ ] GPT2 Classification
  - [ ] Insight Reporting

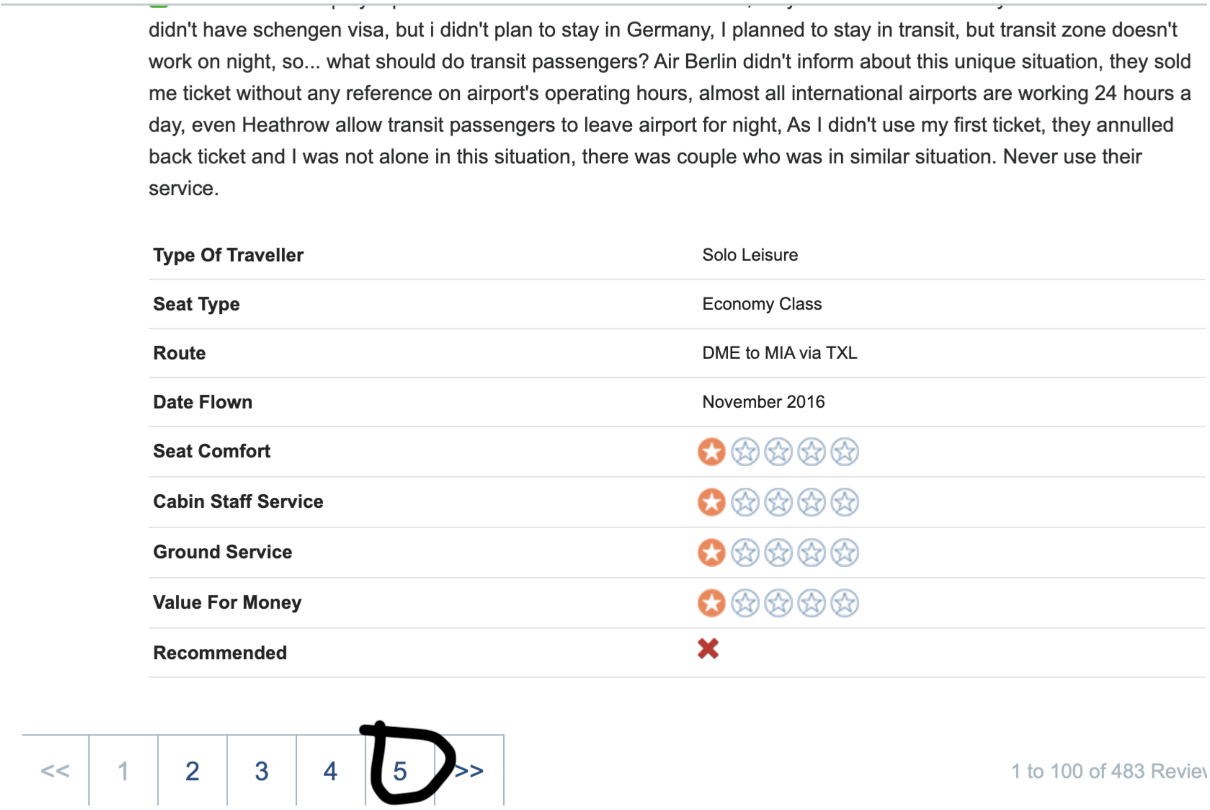
Web Scraping Process

Step 1: Get all the Airline company’s name from A-Z list


```
url='https://www.airlinequality.com/review-pages/a-z-airline-
repsonse = requests.get(url)
soup=BeautifulSoup(repsonse.content,'html.parser')
az_cat=soup.find_all('div',class_='content')
name_list=[]
```

```
for letter in az_cat:
    airlines=letter.find_all('li')
    for airline in airlines:
        name=airline.find('a').get_text(strip=True)
        name_list.append(name)
```

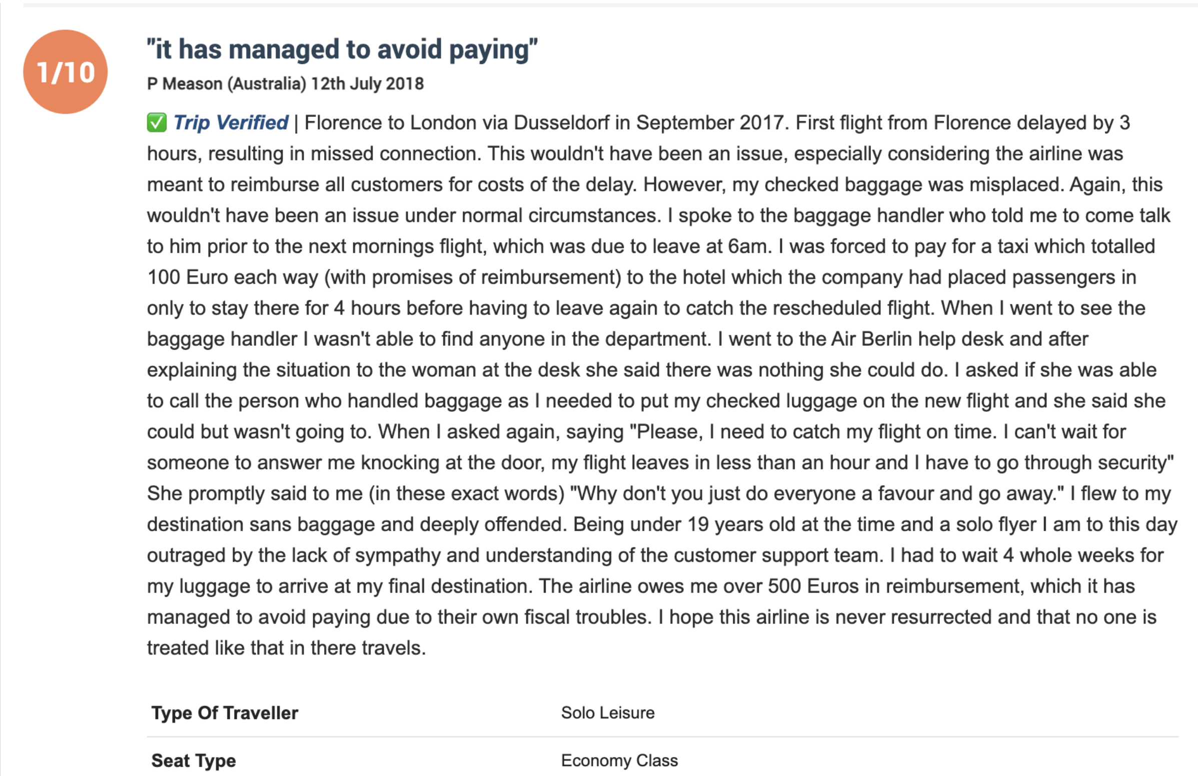
Step 2: Get the total page number for each Airline company and form them as dictionary for easier lookup and iteration




```
def page_scraper(airline_name):
    url='https://www.airlinequality.com/airline-reviews/{}/?s
    try:
        response=requests.get(url.format(airline_name))
        soup=BeautifulSoup(response.content,'html.parser')
        panel=soup.find('article',class_='comp comp_reviews-p
        if panel==None:
            page=1
        else:
            page_list=panel.find_all('a')
            page=page_list[-2].text
```

```
    except requests.RequestException as e:
        print(f'Error fecting the page {airline_name}:{e}')
        page=None
    return page
airline_page={}#set up a empty dic
for airline in url_name_list:
    page=page_scraper(airline)
    airline_page[airline]=page
```

Step 3: Scrapping all useful information from the review block



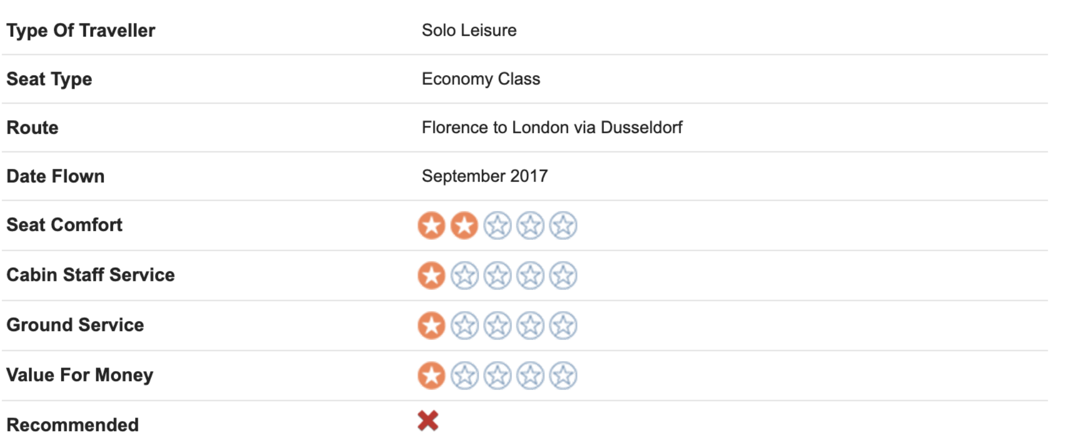
1/10

"it has managed to avoid paying"

P Meason (Australia) 12th July 2018

✅ *Trip Verified* | Florence to London via Dusseldorf in September 2017. First flight from Florence delayed by 3 hours, resulting in missed connection. This wouldn't have been an issue, especially considering the airline was meant to reimburse all customers for costs of the delay. However, my checked baggage was misplaced. Again, this wouldn't have been an issue under normal circumstances. I spoke to the baggage handler who told me to come talk to him prior to the next mornings flight, which was due to leave at 6am. I was forced to pay for a taxi which totalled 100 Euro each way (with promises of reimbursement) to the hotel which the company had placed passengers in only to stay there for 4 hours before having to leave again to catch the rescheduled flight. When I went to see the baggage handler I wasn't able to find anyone in the department. I went to the Air Berlin help desk and after explaining the situation to the woman at the desk she said there was nothing she could do. I asked if she was able to call the person who handled baggage as I needed to put my checked luggage on the new flight and she said she could but wasn't going to. When I asked again, saying "Please, I need to catch my flight on time. I can't wait for someone to answer me knocking at the door, my flight leaves in less than an hour and I have to go through security" She promptly said to me (in these exact words) "Why don't you just do everyone a favour and go away." I flew to my destination sans baggage and deeply offended. Being under 19 years old at the time and a solo flyer I am to this day outraged by the lack of sympathy and understanding of the customer support team. I had to wait 4 whole weeks for my luggage to arrive at my final destination. The airline owes me over 500 Euros in reimbursement, which it has managed to avoid paying due to their own fiscal troubles. I hope this airline is never resurrected and that no one is treated like that in there travels.

Type Of Traveller	Solo Leisure
Seat Type	Economy Class



Type Of Traveller	Solo Leisure
Seat Type	Economy Class
Route	Florence to London via Dusseldorf
Date Flown	September 2017
Seat Comfort	★★☆☆☆
Cabin Staff Service	★☆☆☆☆
Ground Service	★☆☆☆☆
Value For Money	★☆☆☆☆
Recommended	✖

```
def airline_scraper(airline_name,max_page):
    url='https://www.airlinequality.com/airline-reviews/{}/pa
    res=[]
    for i in range(1,int(max_page)+1):
        formatted_url=url.format(airline_name,i)
        response = requests.get(formatted_url)
        soup = BeautifulSoup(response.content, 'html.parser')
        review_list=soup.find_all('article', itemprop='review
        for review in review_list:
            r={}
            r['title']=review.find('h2').get_text(strip=True)
            r['rating']=review.find('span',itemprop='ratingVa
            customer_status=review.find('h3',class_='text_sub
            na_match=re.search(r'\(([^)]+)\)', customer_statu
            if na_match:
                r['nationality']=na_match.group(1)
            date_match=re.search(r'\d+\w*\s+[A-Za-z]+\s+\d{4}
            if date_match:
                r['date']=date_match.group(0)
            r['content']=review.find('div',class_='text_conte
            for tr in review.find_all('tr'):
                detail_name = tr.find('td', class_='review-ra
                detail_value_container = tr.find('td', class_
                if detail_value_container:
                    stars = detail_value_container.find_all('
                    if stars:
                        detail_value = len(stars)
                    else:
                        detail_value = detail_value_container
                    r[detail_name] = detail_value
            res.append(r)
    return res
Final_res_dataframe={}
for airline_name,airline_page in airline_page.items():
    res=airline_scraper(airline_name, airline_page)
    df=pd.DataFrame(res)
```

```
    df['airline']=airline_name
    Final_res_dataframe[airline_name]=df
```

Result: Scraped total 131906 rows of data with 19 features.

Data Cleaning

Step 1: Reviewing Scraping format

airline	title	rating	nationality	date	content	Type Of Trav	Seat Type	Route	Date Flown	Seat Comfor	Cabin Staff S	Food & Beve	Ground Servi	Value For M	Aircraft	Inflight Ente	Wifi & Conne	Recommended
ab-aviation	"pretty decei	9	Netherlands	11th Noveml	,úÖTrip Verif	Solo Leisure	Economy Cla	Moroni to Moheli	Nov-19	4	5	4	4	3				yes
ab-aviation	"Not a good	1	United Kingd	25th June 20	,úÖTrip Verif	Solo Leisure	Economy Cla	Moroni to Anjouan	Jun-19	2	2	1	1	2	E120			no
ab-aviation	"flight was fi	1	United Kingd	25th June 20	,úÖTrip Verif	Solo Leisure	Economy Cla	Anjouan to Dzaoudzi	Jun-19	2	1	1	1	2	Embraer E120			no
adria-airway	"I will never	1	Serbia	28th Septem	Not Verified	Solo Leisure	Economy Cla	Frankfurt to Pristina	Sep-19	1	1		1	1				no
adria-airway	"it ruined ou	1	Netherlands	24th Septem	,úÖTrip Verif	Couple Leisu	Economy Cla	Sofia to Amsterdam via Ljubljana	Sep-19	1	1	1	1	1		1	1	no
adria-airway	"Had very ba	1	Austria	17th Septem	,úÖTrip Verif	Couple Leisu	Economy Cla	Sarajevo to Ljubljana	Sep-19	1	1	1	1	1	CR 900	1	1	no
adria-airway	"worse than	1	Switzerland	6th Septemb	Not Verified	Business	Economy Cla	Ljubljana to Zv°rich	Sep-19	1	1	1	1	1				no
adria-airway	"book anothe	1	Germany	24th August	Not Verified	Solo Leisure	Economy Cla	Timisoara to Munich	Aug-19	1	1	1	1	1	Bombardier	1	1	no
adria-airway	"combined tv	1	Switzerland	6th August 2	,úÖTrip Verif	Solo Leisure	Economy Cla	Pristina to Zv°rich via Ljubliana	Aug-19	1	2	1	1	1		1	1	no
adria-airway	"the crew wa	8	Germany	12th Octobei	,úÖTrip Verif	Family Leisu	Economy Cla	Ljubljana to Munich	Oct-18	4	4	3	5	5				yes
adria-airway	"Very bad ex	1	Germany	5th October	Not Verified	Business	Economy Cla	Zurich to Ljubljana	Oct-18	2	1		1	1		1	1	no
adria-airway	"bad custom	1	United State	29th July 201	,úÖTrip Verif	Family Leisu	Economy Cla	Vienna to Sofia	Jul-18	4	1	1	4	1				no
adria-airway	"overall very	2	France	19th July 201	,úÖTrip Verif	Solo Leisure	Economy Cla	Paris to Skopje via Ljubljana	May-18	3	3		3	2				no
adria-airway	"Would not f	2	Slovenia	30th June 20	,úÖTrip Verif	Business	Economy Cla	Ljubljana to Munich	Jun-18	1	2	2	2	1				no
adria-airway	"very unplea:	3	Czech Repub	24th June 20	,úÖTrip Verif	Couple Leisu	Economy Cla	Ljubljana to Prague	Jun-18	3	3		1	1	A319			no
adria-airway	"Flight was v	10	Slovenia	4th May 201	,úÖTrip Verif	Business	Economy Cla	Frankfurt to Ljubljana	Apr-18	5	5	5	5	5				yes
adria-airway	"delayed for	1	Germany	11th March 2	,úÖTrip Verif	Solo Leisure	Economy Cla	Ljubljana to Frankfurt	Mar-18	2	1	1	1	1		1	1	no
adria-airway	"should be a:	3	United State	5th Decembe	,úÖTrip Verif	Solo Leisure	Economy Cla	Ljubljana to Vienna	Sep-17	2	4	1	1	3	ATR-72			no
adria-airway	"Two nice sh	9	Slovenia	20th Noveml	,úÖTrip Verif	Business	Economy Cla	Ljubljana to Sarajevo	Nov-17	5	5	3	5	3	CRJ700 / ATR72			yes
adria-airway	"extremely b	2	Finland	27th Octobei	,úÖVerified f	Couple Leisu	Economy Cla	Zurich to Ljubjana	Oct-17	3	3		1	1				no
adria-airway	"never fly thi	2	United State	16th Septem	,úÖVerified f	Family Leisu	Economy Cla	Ljubljana to Munich	Jun-17	3	4	1	1	2			1	no
adria-airway	"can't remer	9	Switzerland	19th April 20	,úÖVerified f	Business	Economy Cla	Ljubljana to Zurich	Apr-17	5	5	4	5	4	CR9	4		yes
adria-airway	"seat was qu	8	Austria	27th January	,úÖVerified f	Solo Leisure	Business Cla:	LJU to VIE	Dec-16	5	5	4	5	4	Canadair 70C	5	5	yes
adria-airway	"nice and coi	10	Slovenia	10th Noveml	,úÖVerified f	Business	Economy Cla	LJU to VIE	Oct-16	5	5		5	4	CRJ900			yes
adria-airway	"what a grea	8	Singapore	9th Novembe	,úÖVerified f	Solo Leisure	Business Cla:	LJU to CPH	Nov-16	4	4	3	3	4	CRJ900			yes
adria-airway	"value for m	5	Slovenia	3rd Novembe	,úÖVerified f	Business	Economy Cla	CDG to LJU	Nov-16	3	5	1	4	1		3	3	no
adria-airway	"fleet is tirec	2	Australia	21st October	,úÖVerified f	Solo Leisure	Economy Cla	MUC to LJU	Oct-16	1	1	2	3	1	CRJ-900	1		no
adria-airway	"underwheln	3	Netherlands	10th Octobei	,úÖVerified f	Solo Leisure	Economy Cla	LJU to AMS	Oct-16	3	1	1	2	2	CRJ-900	2		no
adria-airway	"very differe	3	Slovenia	30th Septem	,úÖVerified f	Couple Leisu	Economy Cla	LJU to BRU	Aug-16	2	2		4	4	CRJ900 / A319			no
adria-airway	"staff were a	6	United Kingd	4th Septemb	Booked this f	Family Leisu	Economy Cla	MAN to LJU	Sep-16	2	4	3	3	4	A319	4	4	yes
adria-airway	"Adria do no	3	United Kingd	6th August 2	,úÖVerified f	Couple Leisu	Economy Cla	LGW to LJU	Jul-16	4	5		1	3				no
adria-airway	"Clean and fı	10	Estonia	29th July 201	,úÖVerified f	Solo Leisure	Economy Cla	TLL to ARN	Jul-16	5	5	5	5	5	Canadair 700			yes
adria-airway	"the airline c	8	United State	13th July 201	Flew roundtr	Business	Economy Cla	ZRH to LJU	Jul-16	4	4	3	5	4	CRJ 900			yes
adria-airway	"nice and prc	10	Slovenia	11th July 201	,úÖVerified f	Solo Leisure	Economy Cla	LJU to ZRH	Jul-16	5	5		5	5	CRJ900			yes
adria-airway	"cabin staff ı	6	Poland	10th July 201	Lodz to Paris	Solo Leisure	Economy Cla	LCJ to CDG	Jul-16	3	2	2	4	3	CRJ700	2		yes
adria-airway	"never flying	3	United State	25th January	Adria Airway	Solo Leisure	Economy Cla	MUC to PRN	Jan-16	2	3	1	3	1		1		no

Step2: Dropping unwanted feature

During the First initial investigation on the raw data, I find the Aircraft feature has a lot of missing data and being inconsistent, bring no useful info to this project. So I dropped this feature

Step3: Feature Extraction and Combination

- In the review content, I found the first few words are either Trip Verified or Not Verified, indicating the trip verified status. So I extract a new feature from content and set it as `trip_verified` .
- Combining the title feature with the content feature for better text mining and prediction use.
- The Route feature contains departure city name, arrival city name, and transit city. So I extract three new features departure, arrival, and Flight_type.

```
def Route_extraction(route):
    if pd.isna(route):
```

```
        return {'departure': None, 'arrival': None}

    # Define regex patterns to capture all route formats
    patterns = {
        'transit_to': r'([\w\s]+)\sto\s([\w\s]+)\svia\s([\
        'transit_dash': r'([\w\s]+)-([\w\s]+)\svia\s([\w\s
        'direct_to': r'([\w\s]+)\sto\s([\w\s]+)',
        'direct_dash': r'([\w\s]+)-([\w\s]+)'
    }

    for key, pattern in patterns.items():
        match = re.search(pattern, route)
        if match:
            # Determine if the route is direct or has a tr
            if 'via' in key:
                return {
                    'departure': match.group(1).strip(),
                    'arrival': match.group(2).strip()
                }
            else:
                return {
                    'departure': match.group(1).strip(),
                    'arrival': match.group(2).strip()
                }

    # If no pattern matches, return None for all fields
    return {'departure': None, 'arrival': None}
df['flight_type']=df['Route'].apply(lambda x: 'Transit' if
                                and 'via' in x else 'Direc
```

- Extract two new feature, Month and Year from Date Flown.

Step 4: Reorganizing departure and arrival name

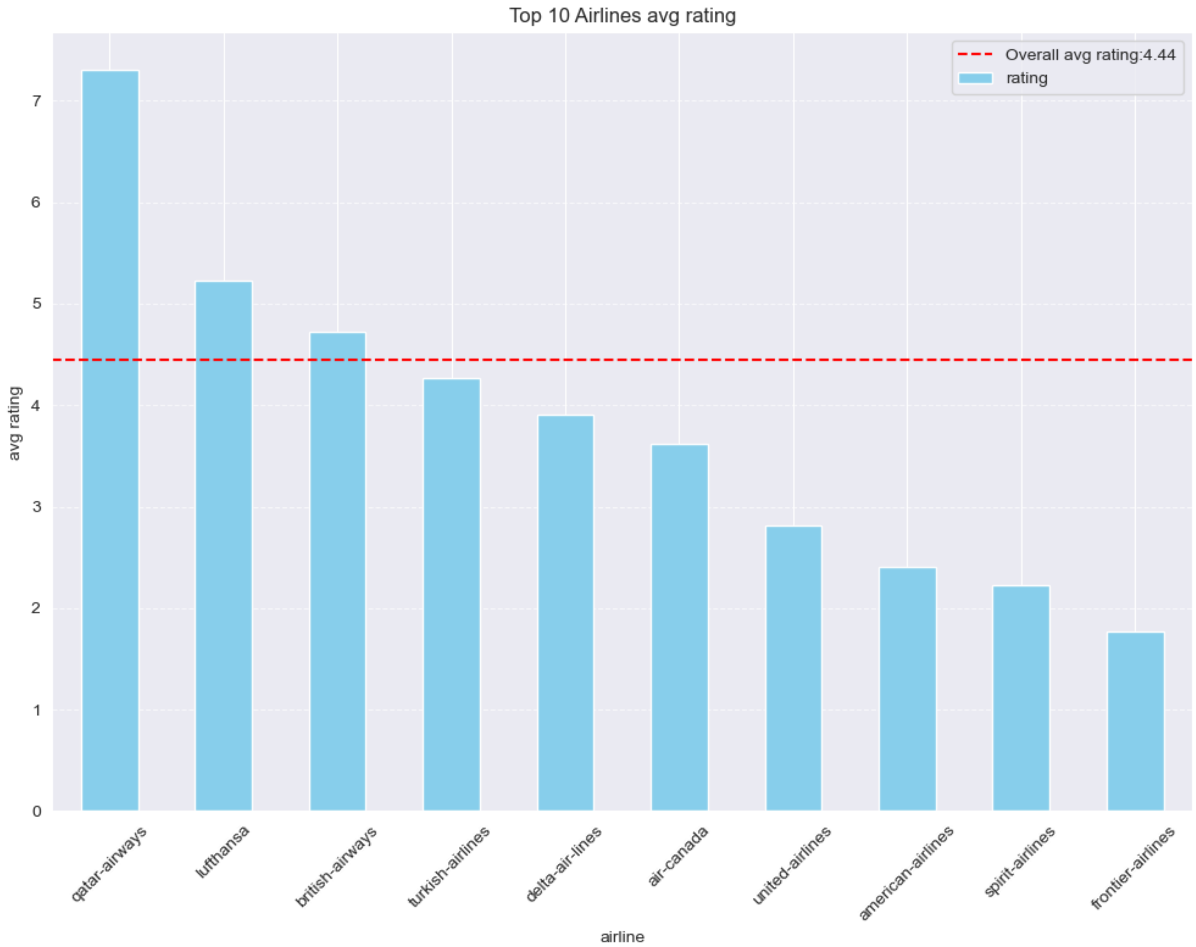
I found the departure and arrival city name very inconsistent and not unify. Eg. NY, New York, JFK, LGA, and JFK NY. So I first filter the city name to get only those name>3, for skipping abbreviation. And use Fuzzywuzzy package to standardize the city name.

```
from fuzzywuzzy import process,fuzz
def stardarize_fuwu(name,standard_names):
    if name is None:
        return None
    match=process.extractOne(name,standard_names,scorer=fuzz.
    if match and match[1]>80:
        return match[0]
    return name
```

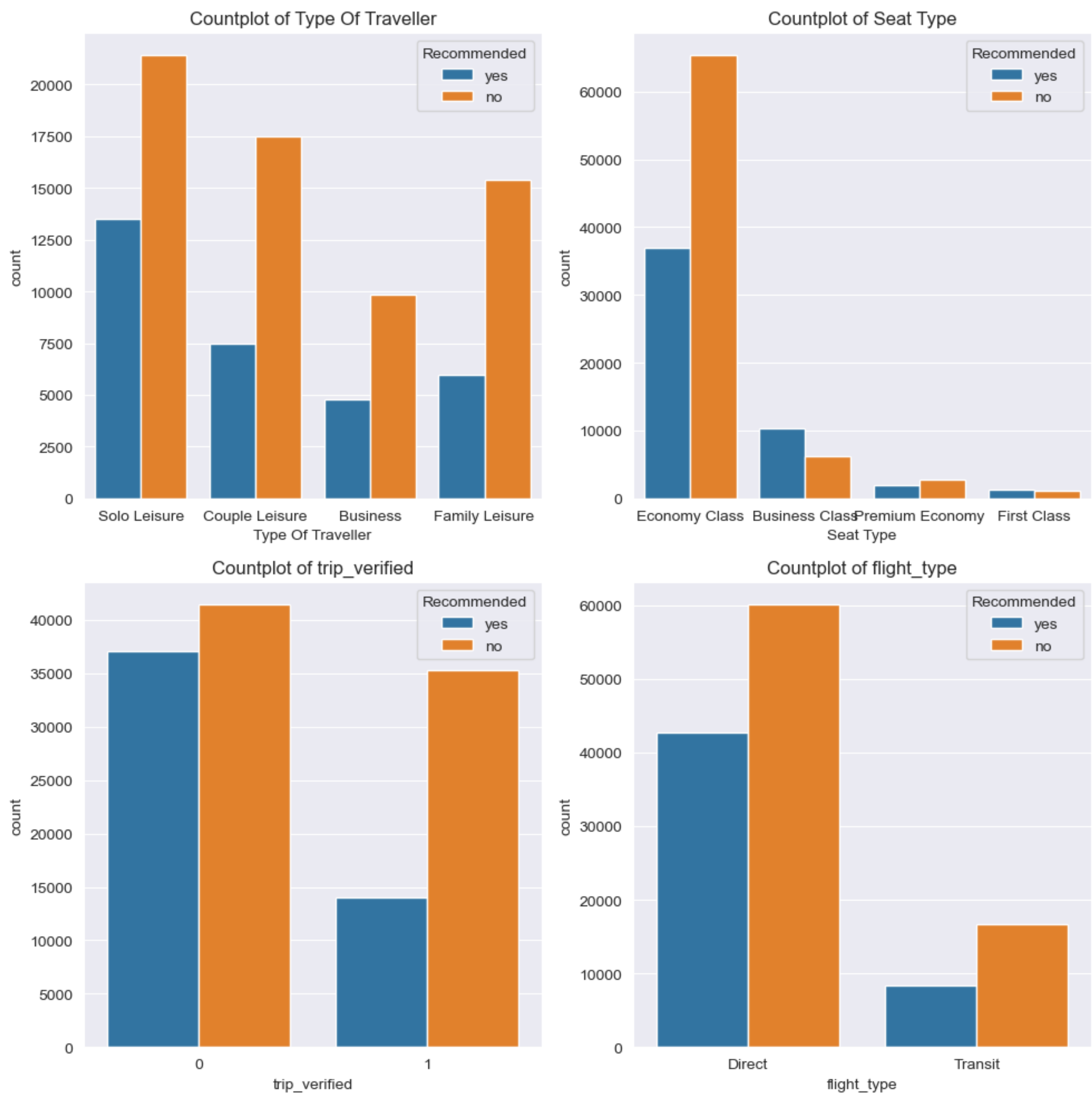
Res: Now we have same amount of data with total of 20 features.

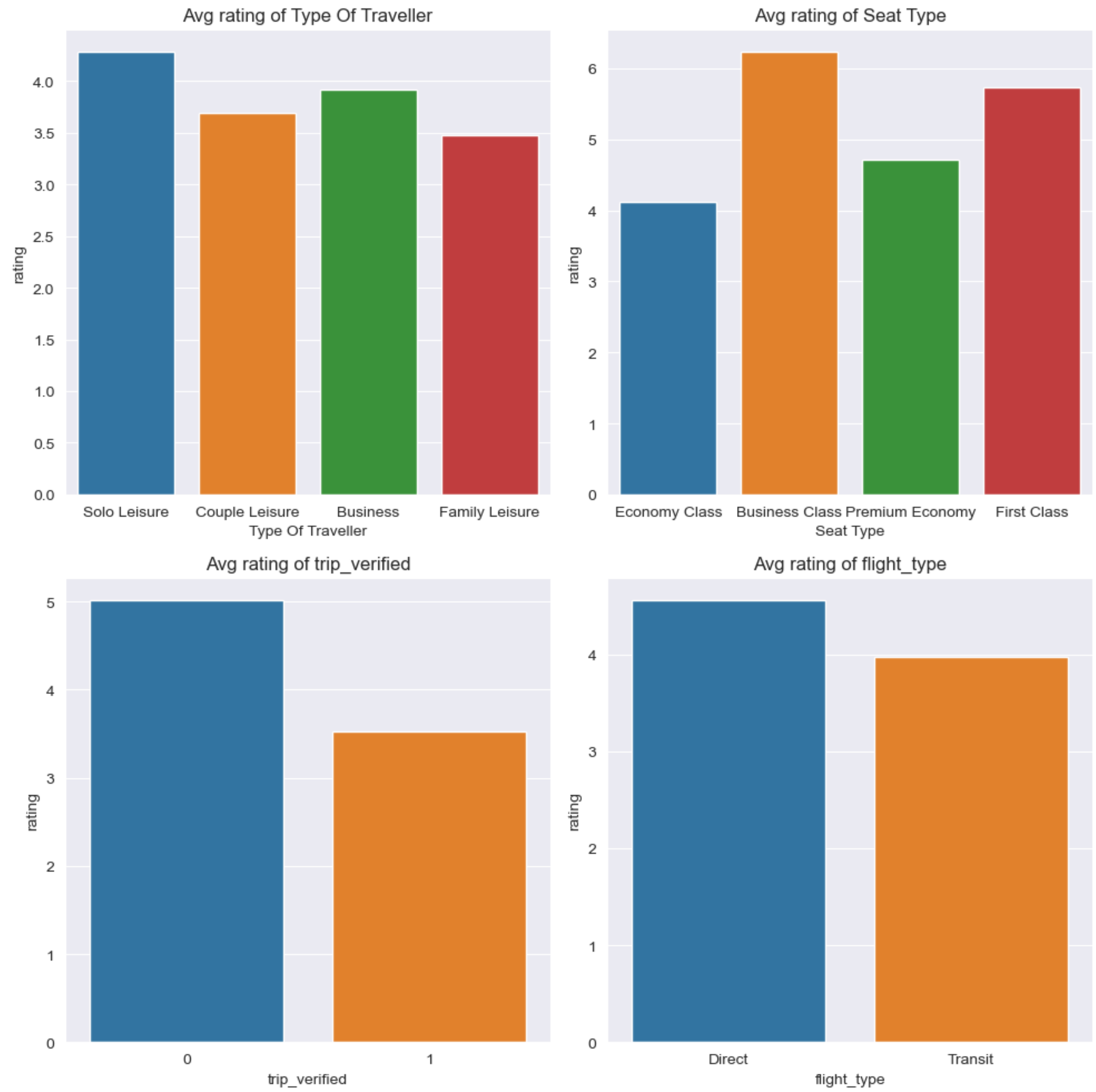
Exploratory Data Analysis

Top 10 Airlines and their Average Rating



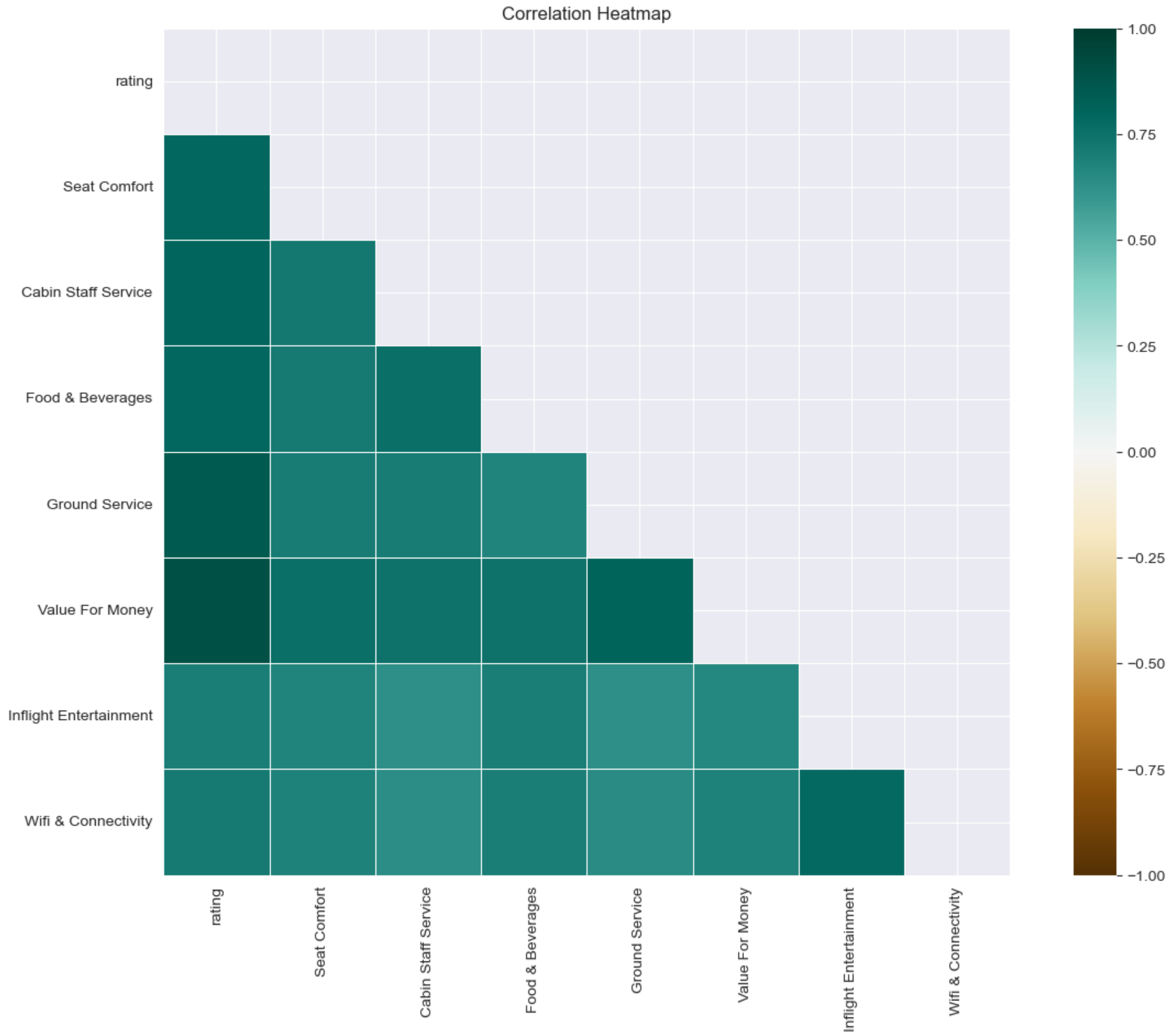
From the chart above, we can see the Qatar airways has much higher rating among all Top 10 volume airline company and spirit and frontier are the lowest two. Making sense due do the reason that Qatar are more higher price ticket than spirit and frontier since they are more like a low-cost airline.





From these two charts, we can see the general recommended ratio and average rating wasn't too high. To be more details, passenger that are taking flight as couple or family that more possible to give no recommend than passenger that are solo or on business trip. The reason for that might be that Couples or families often require more space and comfort during the flight, especially if they have children. If the seating arrangements or amenities are not adequate for their needs, they may feel dissatisfied. Or Families or couples might prioritize different amenities compared to solo or business travelers. If the airline's entertainment options, food, or other amenities are not suitable for their needs or preferences, they may be less likely to recommend the flight. Due to the mentioned reasons, company with more couple/ families type of passenger can be more focus on seating arrangements, entertainment, or space comforting fields.

That's see the correlation between detail part of service and general service.

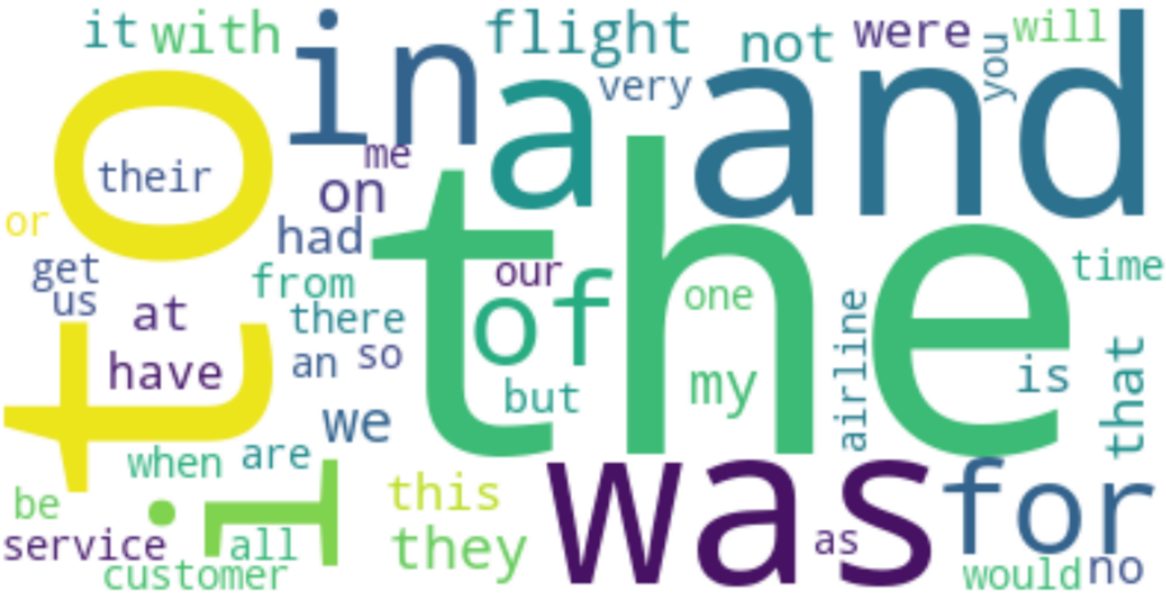


From the heatmap above, we can tell that Value for Money, Ground and Cabin Staff service, and Seat comfort have the stronger positive correlation with the Rating, suggesting that company can work on these part since they might can bring the rating up, please noticed that, however, the correlation doesn't means causation.

Read between the lines. Since we have the text review data, let's see how they can provide any info about how might lead customer to recommend a company or not.

top 50 words for ('yes',)

top 50 words for ('no',)



Above two charts are the word cloud before the text preprocessing. We can see that there aren’t telling us too much info, thus can’t tell how review are difference between recommended or not because there’re lot of stopwords like ‘the’, ‘and’ and ‘to’.

So I add the Lemmatization, punctuation-removing and stopwords-removing to do preprocess on the review text.

```
lemmatizer = WordNetLemmatizer()
def text_preprocessor(text):
    text=re.sub(r'http[s]?://\S+', '', text)
    text=re.sub(r'@\w+','',text)
    text=re.sub(r'#','',text)
    text=re.sub(r'\d+','',text)
    tokens = word_tokenize(text)
    tokens=[word.lower() for word in tokens]
    tokens=[word for word in tokens if word not in string.punc

    stop_words = set(stopwords.words('english'))
    filted_tokens=[word for word in tokens if word not in sto

    lemmatized_tokens=[lemmatizer.lemmatize(token) for token i
    preprocessed_text=' '.join(lemmatized_tokens)
    return preprocessed_text
```

Let’s see if the after-process word cloud is getting better.

top 50 words for ('no',) after preprocessing

Yes! Right now we can tell there's huge difference on both before-after and yes-no comparison. From the word cloud that people who recommended, we can see lots of positive words like 'good', 'excellent', 'great' etc, and some important words like 'seat', 'meal', crew',' service', 'drink', 'food', all these indicating what are some parts that leads to higher probability of recommended outcome. On the other hand, we can see from the word cloud that passenger didn't recommend the flight, some negative words like 'never', 'couldn't', 'worst' etc and import words like 'delay', 'ticket' with 'pay' , 'boarding', 'gate'. These indicating that delay, price value of ticket, and the process of boarding might be the potential reason that leads to negative review and no recommended outcome.

Machine Learning with NLP

The main part of this project is that I want to focusing on comparing different type of NLP technique and see their performance.

Method 1, using traditional Vectorizer package to combine with Classification algorithm. In this method, we apply pipeline to combine the vectorizer and classifier together and iterate each pair to see which has best performance.

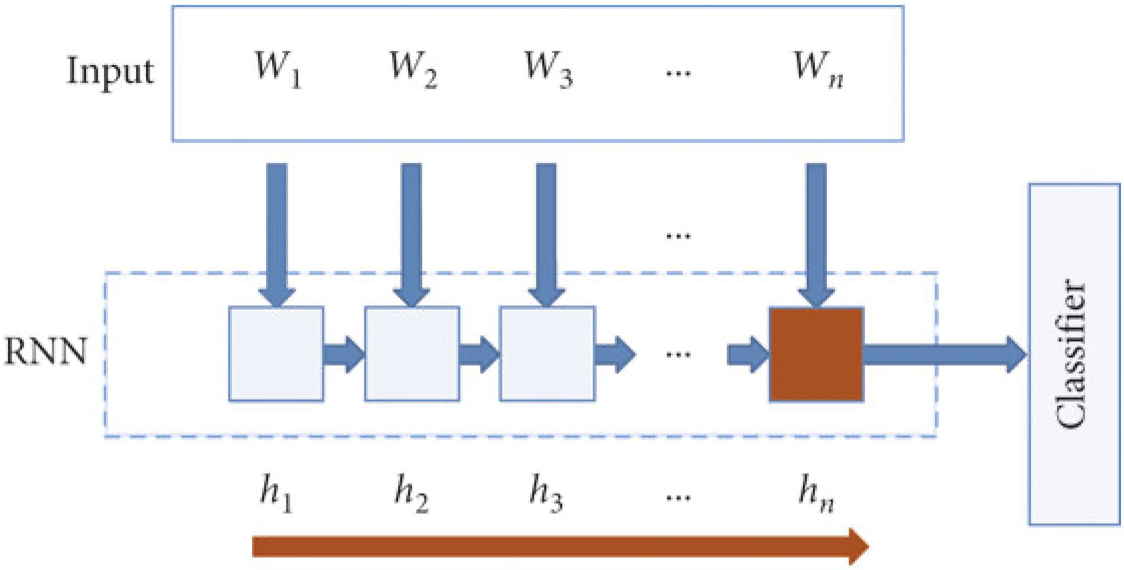
```
combined_vectorizer=FeatureUnion([('tfidf',TfidfVectorizer(mi
max_features=None, strip_accents='unicode', analyzer='word',
ngram_range=(1,3), use_idf=True, sublinear_tf=True, smooth_id
```

```
stop_words='english')),('CountVectorizer',CountVectorizer(min_
max_features=None, strip_accents='unicode', analyzer='word',
ngram_range=(1,3), stop_words='english'))])

classifier={'Naive Bayes':MultinomialNB(),
            'Logistic Regression': LogisticRegression(max_ite
            'Random Forest': RandomForestClassifier()
}

vectorizer = {
    'tfidf': TfidfVectorizer(min_df=3, max_features=None, str
    'CountVectorizer': CountVectorizer(min_df=3, max_features=
    'Combined': combined_vectorizer
}

MLresult=[]
for vec_name, vec in vectorizer.items():
    for clf_name,clf in classifier.items():
        pipe=Pipeline([
            (vec_name,vec),
            (clf_name,clf)
        ])
        pipe.fit(X_train,y_train)
        pred=pipe.predict(X_valid)
        accuracy=accuracy_score(y_valid,pred)
        MLresult.append({
            'Classifier': clf_name,
            'Vectorizer': vec_name,
            'Accuracy':accuracy
        })
```

	Classifier	Vectorizer	Accuracy
0	Naive Bayes	tfidf	0.885452
1	Logistic Regression	tfidf	0.921954
2	Random Forest	tfidf	0.894284
3	Naive Bayes	CountVectorizer	0.869911
4	Logistic Regression	CountVectorizer	0.923357
5	Random Forest	CountVectorizer	0.891896
6	Naive Bayes	Combined	0.873323
7	Logistic Regression	Combined	0.923357
8	Random Forest	Combined	0.894360

From the result, we can see Logistic Regression with Count Vectorizer perform the best accuracy among all other pair. And I suggested that this method is just a sample that can be refined by add more classifier like SVM, Gradient Boost, or Decision Tree etc with cross validation method and grid search to get more better performance.

Method 2, Simple RNN is a type of neural network particularly useful for processing sequences of data. In the context of text classification, RNNs can analyze and understand the sequential nature of text data, making them suitable for tasks such as sentiment analysis, spam detection, or language translation.



In my coding, I build simple-RNN by using Keras API, preprocesses the data, defines the model architecture, compiles the model, and trains it on the provided data.

The process is from Tokenization, Padding Sequences, Define Model, Model Compilation, Model Training to Prediction. The result of training and prediction we get is :

Training

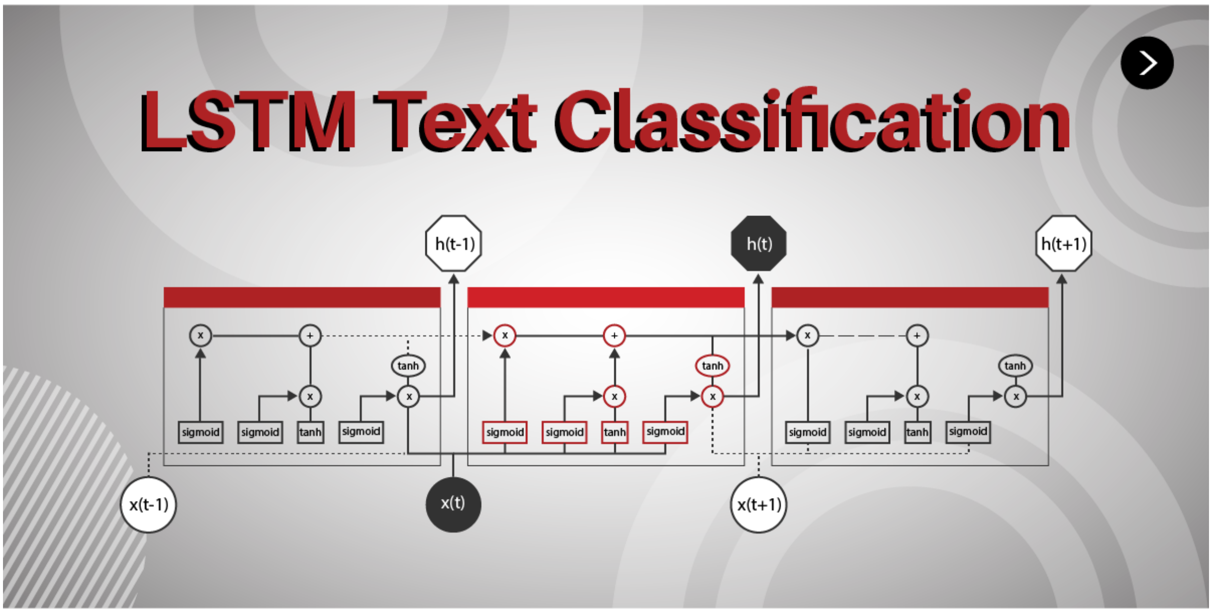
Epoch 1/2
loss: 0.5568 - accuracy: 0.7099
Epoch 2/2
loss: 0.4635 - accuracy: 0.7855

Prediction

Accuracy: 81.02%

Method 3 LSTM Long Short-Term Memory is a type of recurrent neural network (RNN) architecture that is capable of learning long-term dependencies in sequential data. It's particularly effective in tasks where context over longer sequences is important, such as language modeling, text generation, and speech recognition.

LSTM networks have a unique structure of memory cells and gates that regulate the flow of information through the cell. These gates (input, forget, and output gates) help the LSTM to selectively remember or forget information over time, which enables it to handle vanishing gradient problems and capture long-range dependencies more effectively compared to traditional RNNs.



Since we already build the RNN model, all we need to do is to add LSTM layer when doing the Model Definition and change the optimizer from Adam to RMSprop. Then do the training and prediction again. The result:

Training

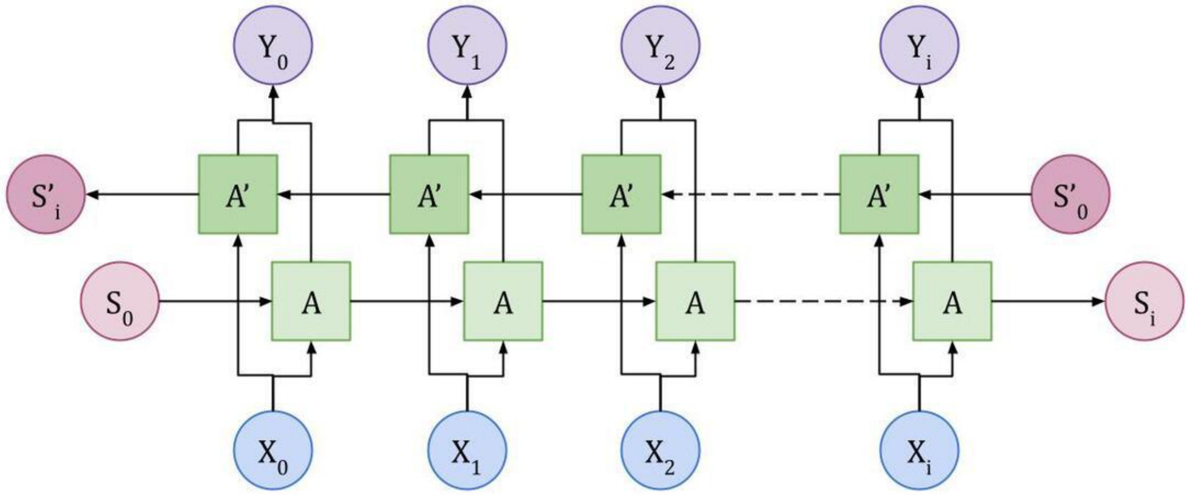
loss: 0.2671 - accuracy: 0.8970
Epoch 2/2
loss: 0.2109 - accuracy: 0.9246

Prediction

Accuracy: 97.29%

Method 3, the Bidirectional RNN is a type of recurrent neural network architecture that processes input sequences in both forward and backward directions. This allows the model to capture patterns and dependencies from both past and future contexts, which can be particularly useful in tasks where understanding the entire sequence is important, such as machine translation, speech recognition, and sentiment analysis.

Bidirectional RNNs consist of two separate recurrent layers: one processes the input sequence in a forward direction, while the other processes it in a backward direction. The outputs from both directions are typically concatenated or combined in some way to produce the final output.



Still, since we already build the LSTM RNN model, all we need to do is to wrap up the LSTM by using Bidirectional layer. The training and prediction result

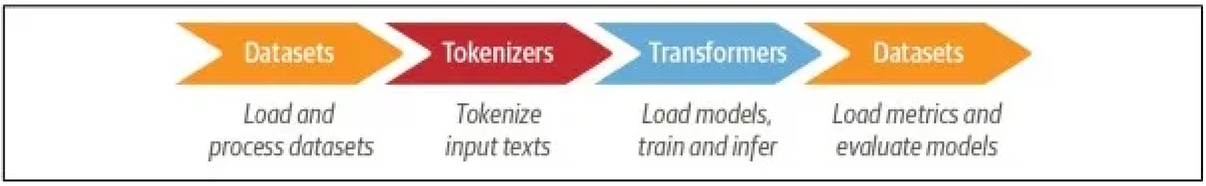
Training

Epoch 1/2
loss: 0.6684 - accuracy: 0.6124
Epoch 2/2
loss: 0.6682 - accuracy: 0.6124
Prediction

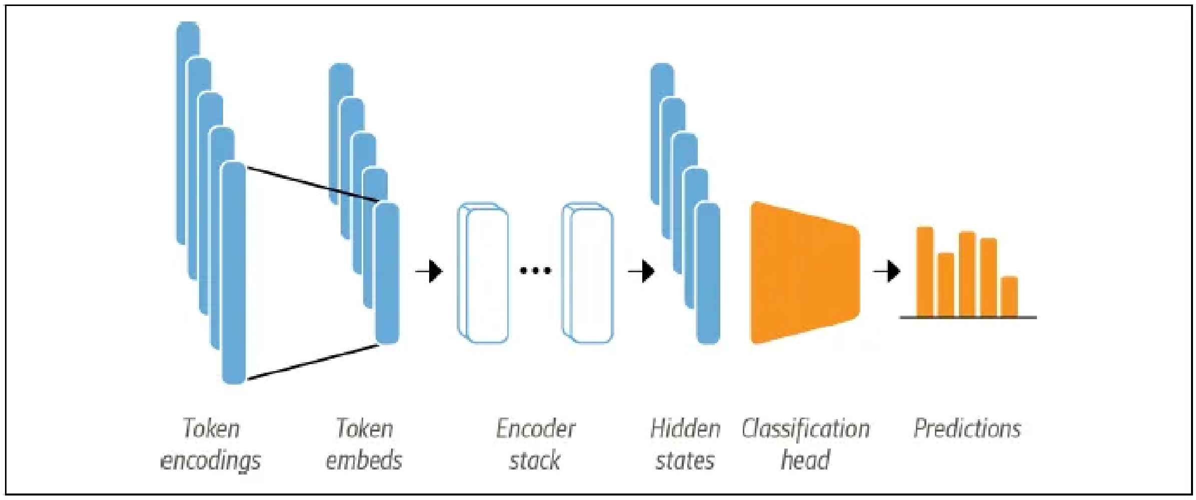
Accuracy:50.00%

Method 4, The Transformer is a type of deep learning model that revolutionized natural language processing (NLP) tasks. Unlike traditional sequential models like RNNs and LSTMs, transformers process entire sequences of data simultaneously. They utilize self-attention mechanisms to weigh the importance of different words in a sequence, allowing them to capture long-range dependencies efficiently. Transformers consist of an encoder-decoder architecture, but for text classification tasks, you typically only need the encoder part.

This is a broad view of the whole process



Detailed view



In my coding, firstly is to build the custom layers TransformerEncoder and TokenAndPositionEmbedding for the first part of the process I showing above. defined the input layer then pass thru the TokenAndPositionEmbedding function then use the TransformerEncoder to process the embedded sequences. GlobalAveragePooling1d is to aggregate info from all tokens into a fixed repr. Regularization is applied to prevent overfitting. Dense layer with sigmoid activation to produce the binary classification output.

There's my training and predicting result of Transformers:

Training

Epoch 1/5
loss: 0.3350 - accuracy: 0.8429 - val_loss: 0.2080 - val_accuracy: 0.9205
Epoch 2/5
loss: 0.1963 - accuracy: 0.9275 - val_loss: 0.2128 - val_accuracy: 0.9246
Epoch 2: early stopping

Prediction

Accuracy:97.43%

Now let's compare the result of all common and advanced language prediction model to see who did it better.

	Model	Accuracy
0	Simple RNN	0.8102
1	LSTM	0.9729
2	BiDirectional RNN	0.5000
3	Transformer	0.9743

We can see Transformer and LSTM are perform much better than the other two model. However, I think if spend more time on the fine-tunning. All model can be level up to perform a much more precise prediction.

GPT2-Classification

When I was doing research on the text classification, I noticed there's one model is very interest that can be the base of LLM technique. That is the fine-tune GPT2 model, following link is the reference website that tutorial of this method on GitHub: 🎱 **GPT2 For Text Classification using Hugging Face 🤗 Transformers** . In this project, I tried to use GPT2 model with the Huggingface Transformers library on my dataset. With the constraint and limitation of RAM of Google Colab, I cannot perform the whole dataset (not even 1/10 of the dataset actually). So this part is only performing a sample data (500 rows) on the model since the more important is I want to introduce this model and method.

Main idea of this model is that, GPT2 is a decoder transformer, making the last token of the input sequence contains all the info needed for prediction. So we use the info to make prediction instead of generation, GPT is stand for generative pre-trained.

To be more easily understand, we firstly build the Pytorch Dataset class, load the dataset where texts are labeled for classification, create Gpt2ClassificationCollator to dynamically pads the batched data to uniform length, ensuring efficient training. Then implement iteration to train the model with customer Train function and evaluate its performance by customer Validation function.

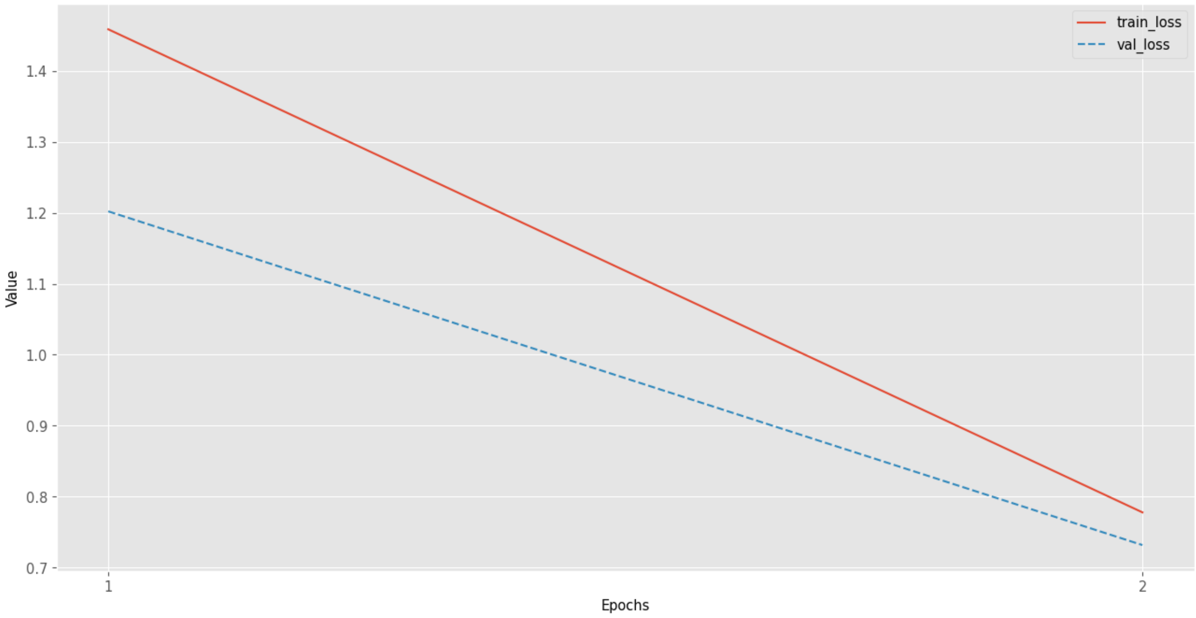
Then final training results are:

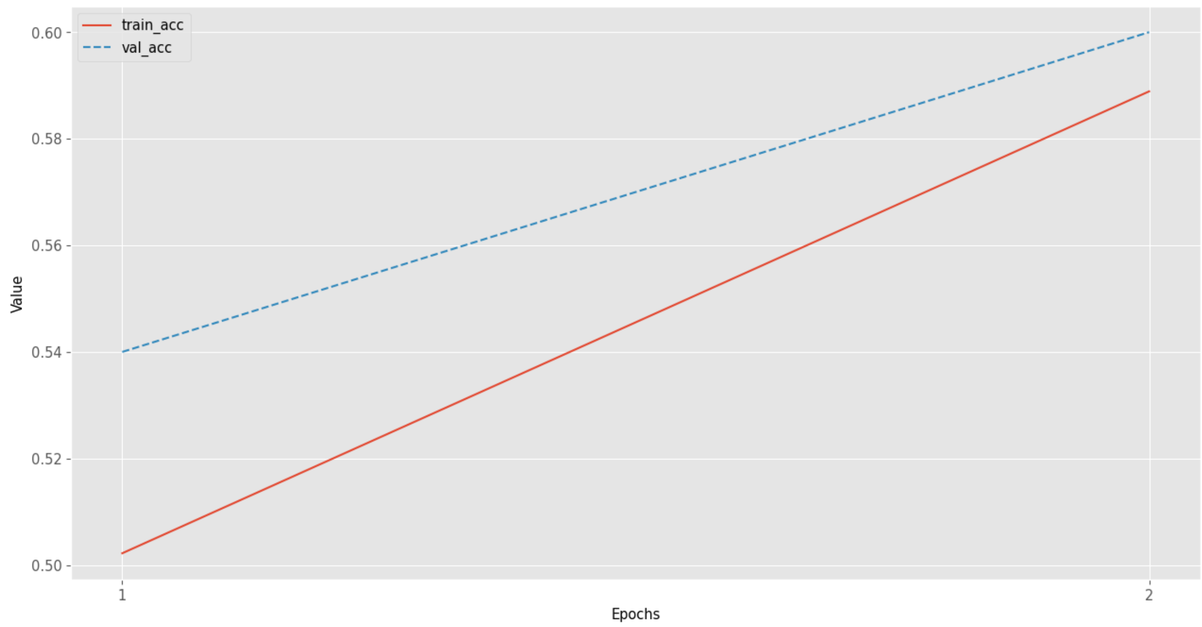
Epoch 1/2

train_loss: 1.45844 - val_loss: 1.20198 - train_acc: 0.50222 - valid_acc: 0.54000

Epoch 2/2

train_loss: 0.77794 - val_loss: 0.73191 - train_acc: 0.58889 - valid_acc: 0.60000





Due to the limitation of RAM of the Google Colab, it’s hardly to perform too much epochs and size of dataset. But we still can see some potential of this method on the text classification. There’re getting more and more advanced method pop out everyday, however, knowing how the traditional model is working would be the best to understand the foundation of this huge topic.

Final Wrap-Up

In this report, we identified include service quality, seating comfort, value for money, ground and cabin staff service these indicators can be important affect to the overall customer satisfaction. Using the machine learning model that analyzes text from the customer feedback to successfully predict the whether a customer would recommend the flight. From these findings, airline company can find where they can enhance their service quality and operational efficiency. Even if satisfaction ratings are available, review content offers deeper insights into specific aspects of the customer experience, highlighting areas for potential improvement and providing qualitative data that complements quantitative ratings.

Conclusion

The project demonstrated the potential of advanced NLP methods in extracting meaningful insights from customer reviews, which could help airlines improve their services and customer satisfaction. The exploration of GPT-2, despite limitations due to computational resources, highlighted its capabilities in generative tasks adapted for classification.

This project also underscores the importance of continuous model refinement and the potential of integrating more comprehensive datasets and advanced techniques to enhance predictive accuracy and business insights.