Project 3: Web APIs & NLP

Lo Kok Fu

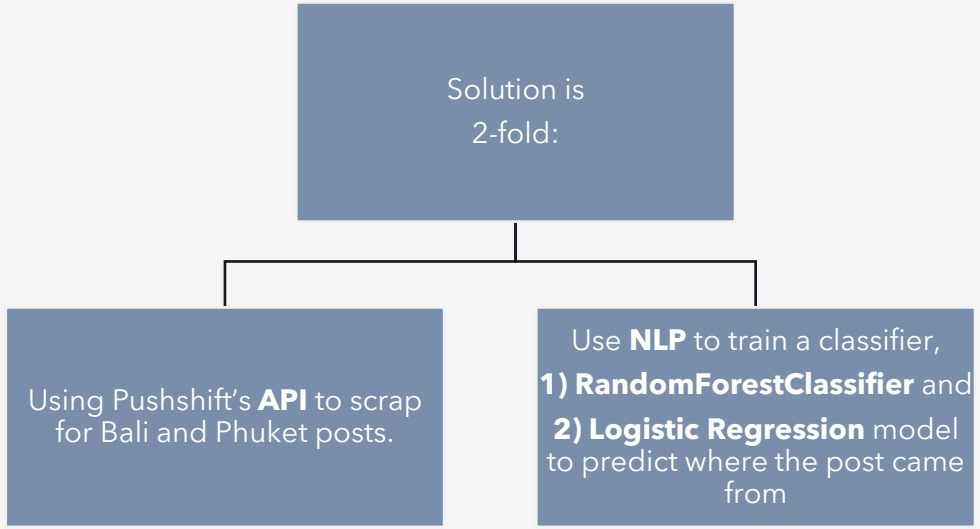
DSIF 2

Agenda

- Problem Statement
- Introduction
- Obtaining Data
- Data Exploration
- Data Processing / Engineering
- Modelling
- Results
- Recommendations

Problem Statement

Utilize NLP to train a classifier on a binary classification problem to predict which subreddit post it comes from. Gather our data through webscrapping from subreddit using API.



GENERAL ASSEMBLY

Introduction

- Topic classifier: Tourist beaches.
- Bali & Phuket
- Similar point of Interest
- Familiarity
- COVID-19. Let's see what we scrape

```python
# create function for web scrapping submissions:

def scrapping(url, loops, subreddit):
    df = []
    start_time = time.time() # get the time in seconds since epoch
    params = {'subreddit': subreddit,
    'size': 100,
    'before': round(start_time)
    }

    for i in range(loops):
        current_time = time.time()
        #requesting data
        res = requests.get(url, params)
        print(f'res status {i+1}: ', res.status_code)

        data = res.json()
        posts = data['data']
        post_df = pd.DataFrame(posts)
        df.append(post_df)
        #get oldest post time and use as before parameter in next request
        old = post_df['created_utc'].min()
        params['before'] = old
        time.sleep(1)
        reddit_posts = pd.concat(df)

        filename = subreddit + '_submission.csv'

    return reddit_posts.to_csv('./datasets/' + filename, index=False)
```

```python
# scrape for bali submission data set from Reddit.
url = 'https://api.pushshift.io/reddit/submission/search'
loops = 8 # no. of loops to scrap
subreddit = 'bali' # subreddit topic

scrapping(url, loops, subreddit)
```

```
res status 1:  200
res status 2:  200
res status 3:  200
res status 4:  200
res status 5:  200
res status 6:  200
res status 7:  200
res status 8:  200
```

```python
bali_submission = pd.read_csv('./datasets/bali_submission.csv')
```

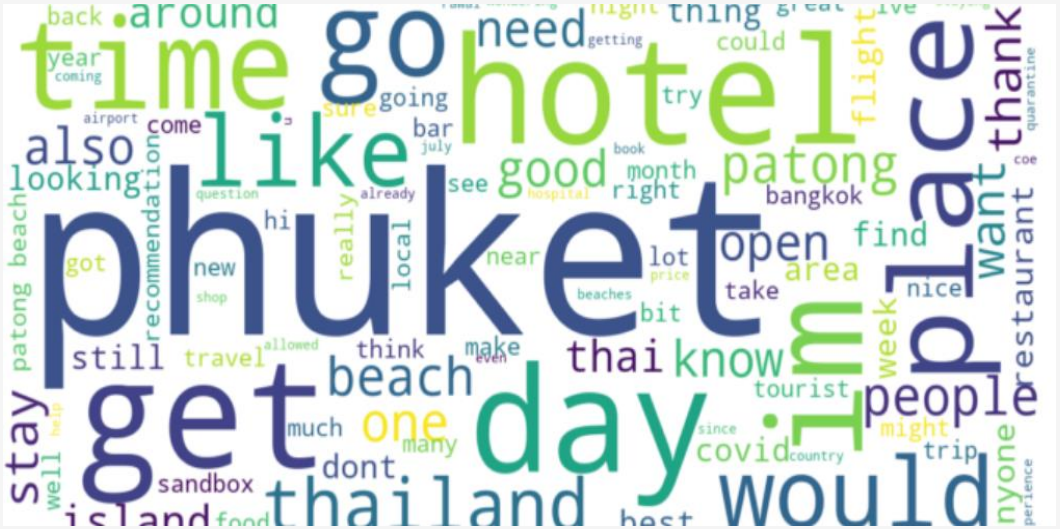
Obtaining Data

- Subreddit limited scrapping no.
- Created a function to loop scrapping
- Looping from oldest post scrapped
- Concat
- File save
- Data Cleaning
  - Stop words
  - Punctuations
  - Lower case
  - Removed duplicates
  - Removed int
  - Filled in null values
  - Merged 'self_text' & 'title
  - Concat comments to df
- 1600 for each topic , Total 3200
- Balance: 3075
- Data imbalance

```
1    0.505691
0    0.494309
```

GENERAL ASSEMBLY

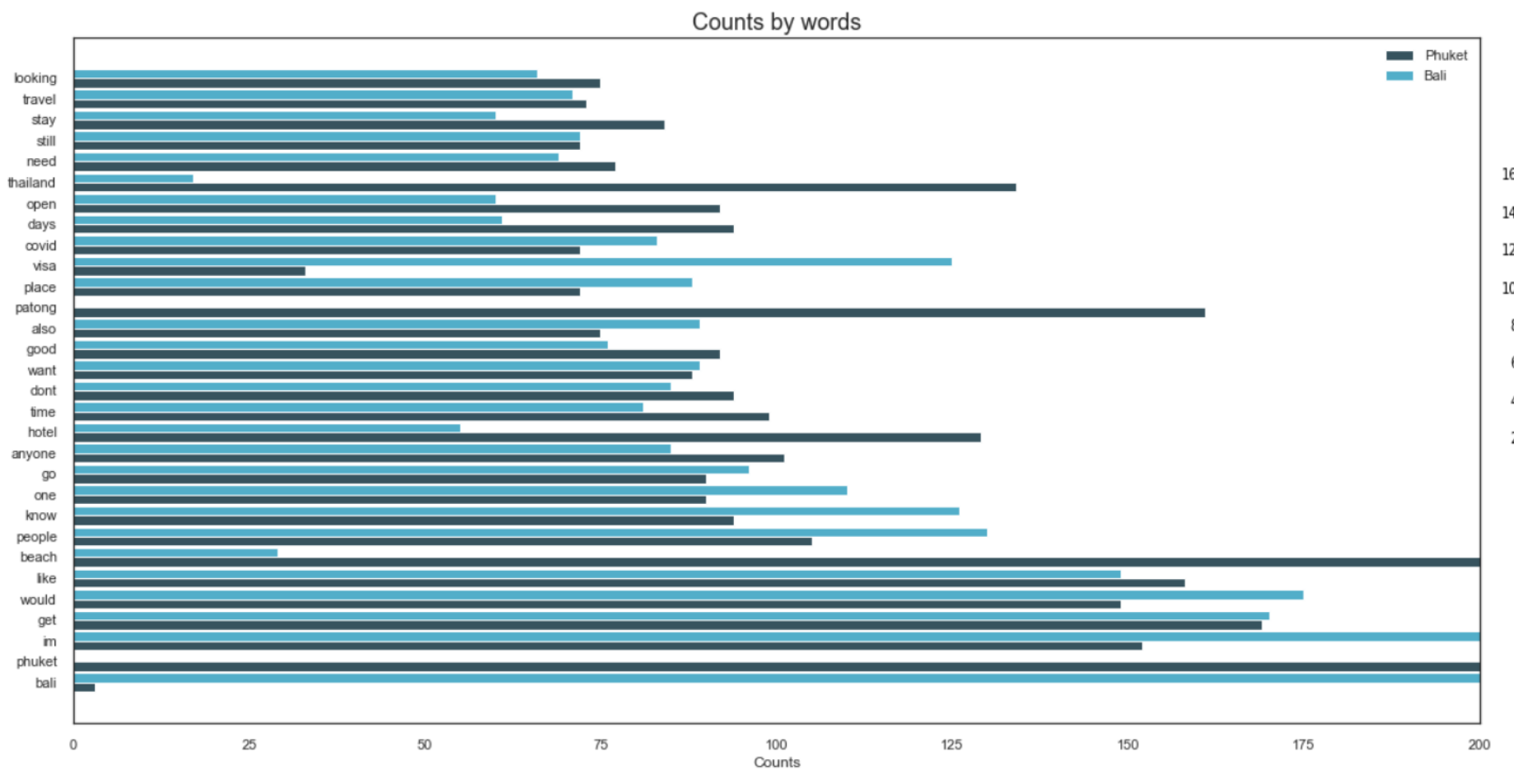
Data Exploration

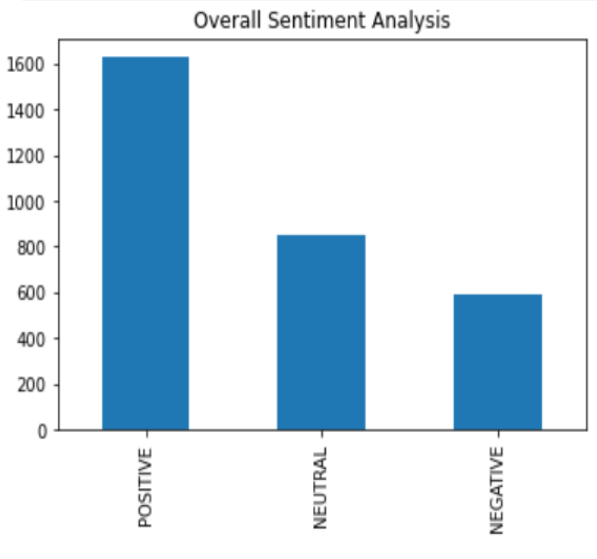


- 'Quarantine'& 'Jakarta'. This may be due to essential visitors are to serve quarantine in Jakarta before visiting bali.
- 'Villa'. Holiday villas are spruced all over the island. thus the frequent mentioned of it.
- 'Visa'. Foreign Visitors to Bali will require visas.

- 'Patong' is a name of a place/beach in phuket
- 'hotel', Recommendations?
- 'Bangkok' the capital of thailand.
- 'sandbox' linked to Covid-19
- 'open'. Opening of Borders?

Data Exploration





GENERAL ASSEMBLY

Data Processing / Engineering

- Word Count
- Sentiments values (Vader)
- Function transformer
  - Standard Scaler
  - Count vectorizer
- Pipeline to combine features and estimator

```
                ne  to combine features and estimator
            eline([
        ures', FeatureUnion([('numeric_features', Pipeline([('selector', get_numeric_d
           ('text_features', Pipeline([('selector', get_text_data),('cvec', CountVectorize
    'rf', RandomForestClassifier())


params = { 'rf__n_estimators': [80,90,110,120],
           'rf__max_depth':[5,10,15,20],
            'features__text_features__cvec__stop_words' : [None, stop]}


gs_rf = GridSearchCV(pipe_rf, params, cv=5, verbose = 1, n_jobs = -1)

gs_rf.fit(X_train, y_train)
```

GENERAL ASSEMBLY

Modelling

- Random Forest Classifier
  - Gridsearch CV with Hyperparameter Tuning:
    - n_estimators = 110
    - max_depth = 20
- Logistic Regression
  - Gridsearch CV with Hyperparameter Tuning:
    - Penalty = l2
    - Solver = 'sag'
    - C value = 10

Modelling 1: RandomForestClassifier

```
# utilising pipline  to combine features and estimator
pipe_rf = Pipeline([
    ('features', FeatureUnion([('numeric_features', Pipeline([('selector', get_numeric_data) ])),
            ('text_features', Pipeline([('selector', get_text_data),('cvec', CountVectorizer()) ]))])),
    ('rf', RandomForestClassifier())
])

params = { 'rf__n_estimators': [80,90,110,120],
          'rf__max_depth':[5,10,15,20],
           'features__text_features__cvec__stop_words' : [None, stop]}


gs_rf = GridSearchCV(pipe_rf, params, cv=5, verbose = 1, n_jobs = -1)

gs_rf.fit(X_train, y_train)
```

Modelling 2: Logistic Regression

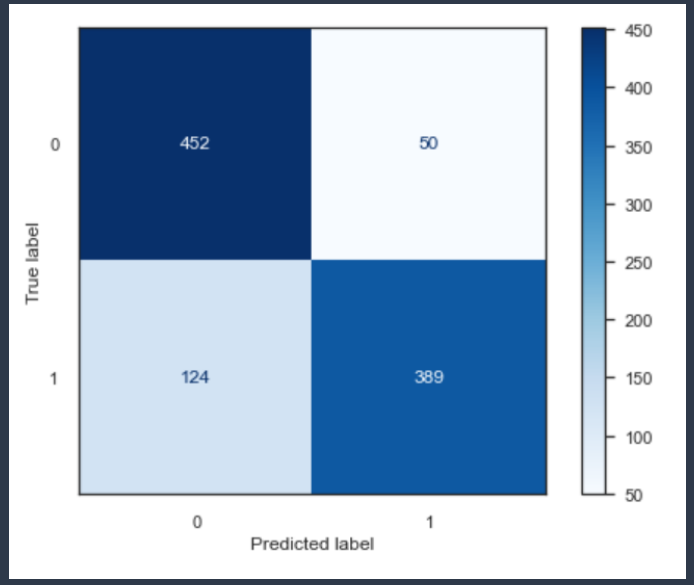
```
pipe = Pipeline([
    ('features', FeatureUnion([
            ('numeric_features', Pipeline([
                ('selector', get_numeric_data),
                ('ss', StandardScaler())
            ])),
            ('text_features', Pipeline([
                ('selector', get_text_data),
                ('cvec', CountVectorizer())
            ]))
    ])),
    ('logreg', LogisticRegression())
])

params = {
            'features__text_features__cvec__stop_words' : [None, stop],
            'logreg__penalty' : ['l1', 'l2'],
            'logreg__solver': ['newton-cg', 'lbfgs', 'liblinear', 'sag', 'saga'],
            'logreg__C' : [100, 10, 1.0, 0.1, 0.01]

}

gs = GridSearchCV(pipe, params, cv=5, verbose = 1, n_jobs = -1)

gs.fit(X_train, y_train)

```

```
Fitting 5 folds for each of 100 candidates, totalling 500 fits
```

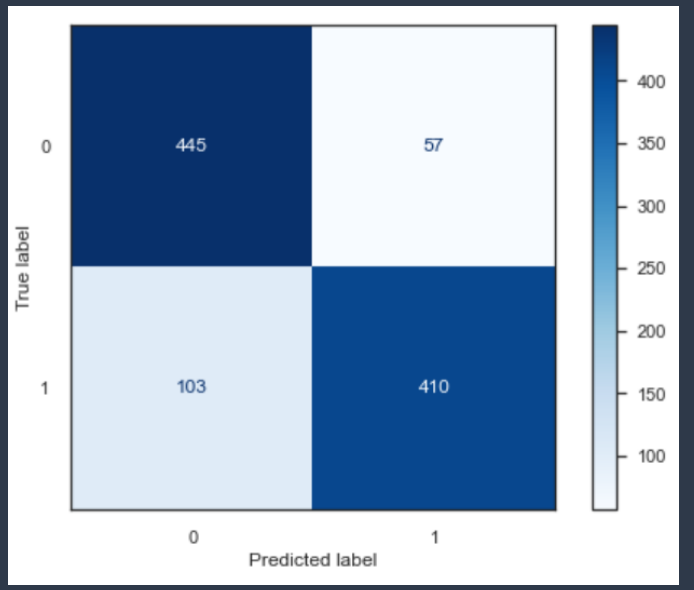
GENERAL ASSEMBLY

Results

Model	Train Score	Test Score	Specificity	F1 Score	AUC
Random Forest Classifier	0.92718	0.82856	0.90039	0.81722	0.94435
Logistic Regression	0.94611	0.84236	0.8864	0.83673	0.93579

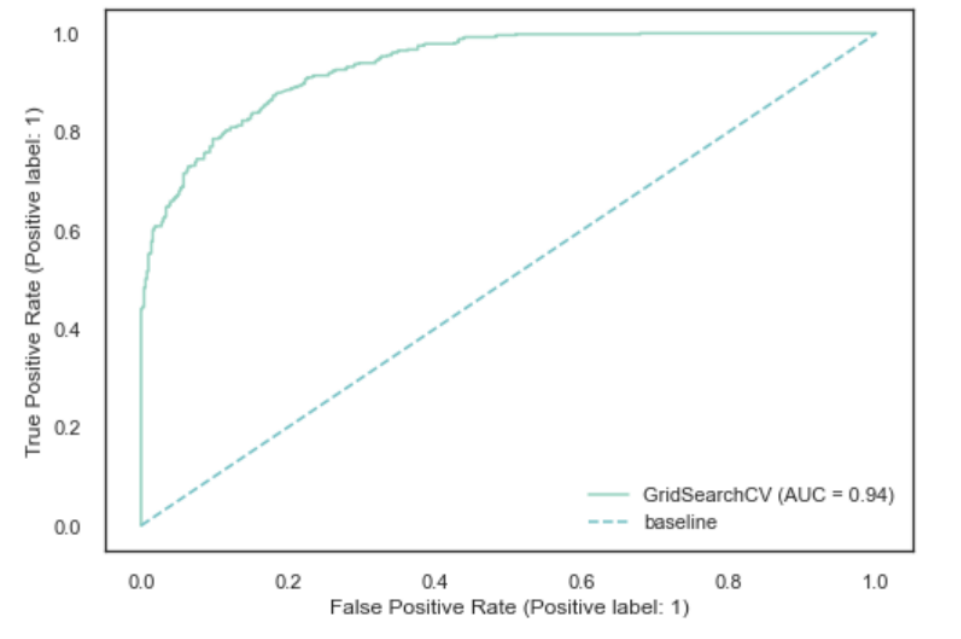


Random Forest Classifier Confusion Matrix

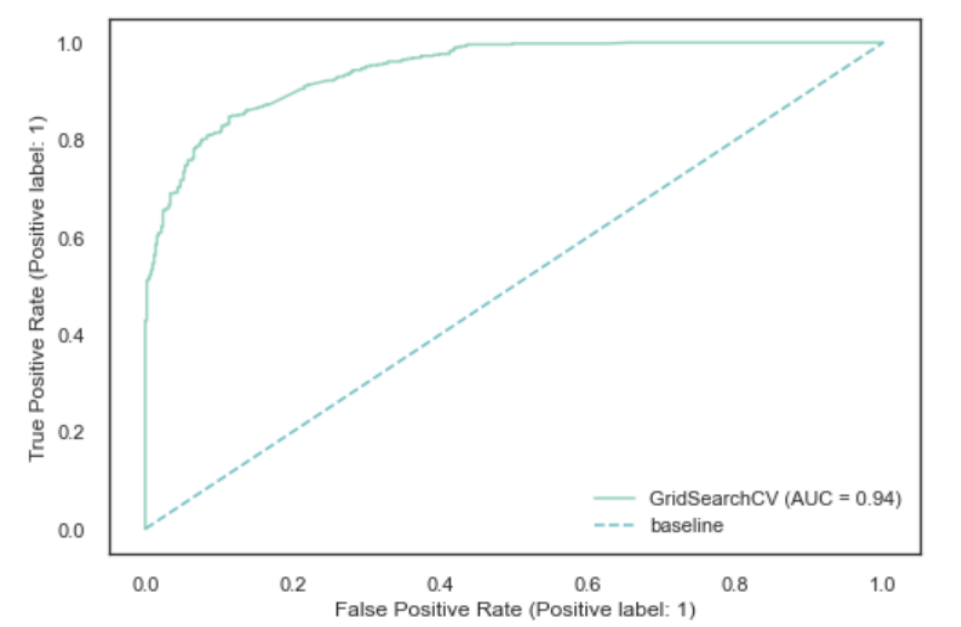


Logistic Regression Confusion Matrix

GENERAL ASSEMBLY



Logistic Regression ROC Curve



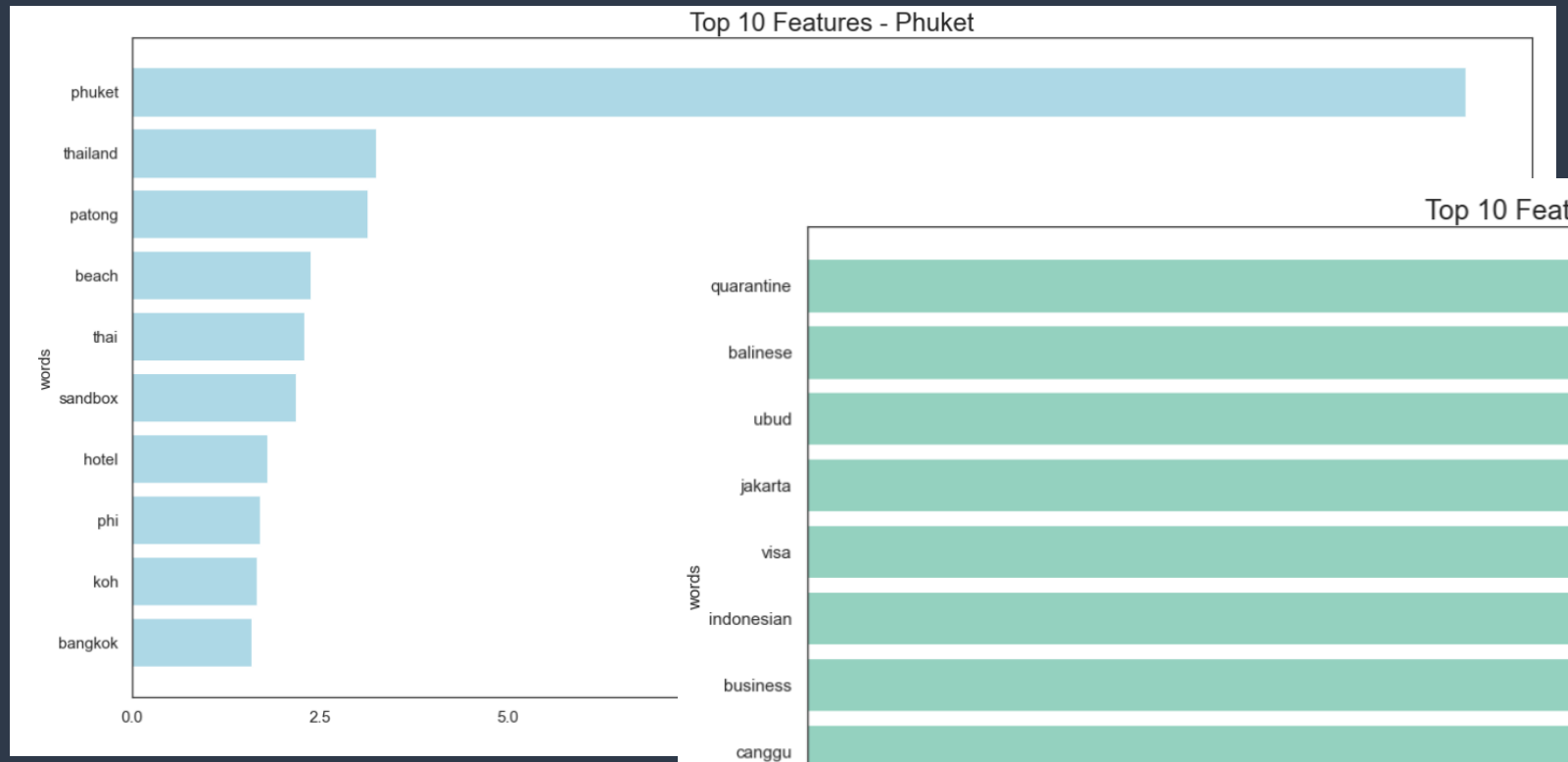
Random Forest ROC Curve

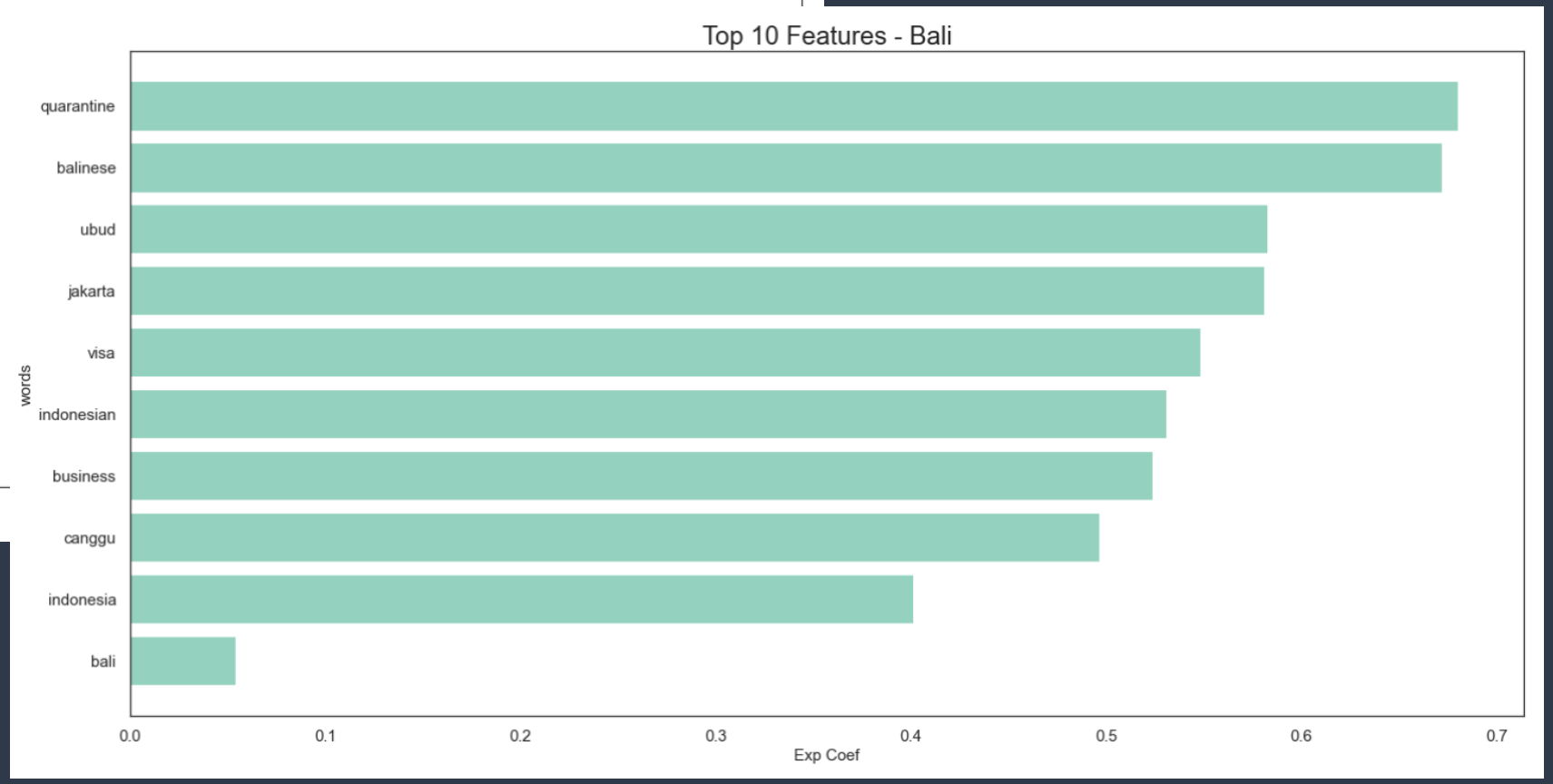
Results ROC Curve

Model	Train Score	Test Score	Specificity	F1 Score	AUC
Random Forest Classifier	0.92718	0.82856	0.90039	0.81722	0.94435
Logistic Regression	0.94611	0.84236	0.8864	0.83673	0.93579

GENERAL ASSEMBLY

Top 10 Features





Recommendations

- Logistic Regression
  - Best F1 scoring
  - Accuracy Score
  - Slight difference in scores
- More time, Faster CPU
  - Explore hyperparameter Tuning
  - Feature engineering
  - TFIDF
  - Other models

Thank you

GENERAL ASSEMBLY