SENTIMENT ANALYSIS OF OPAY APP REVIEWS

- BY: GBADAMOSI OLUWASEYI EMMANUEL
- FINAL YEAR PROJECT

INTRODUCTION

Sentiment analysis, is an approach to natural language processing (NLP) that identifies the emotional tone behind a body of text. This is a popular way for organizations to determine and categorize opinions about a product, service or idea. It involves the use of data mining, machine learning (ML) and artificial intelligence (AI) to mine text for sentiment and subjective information. In recent years, several methods of sentiment analysis have been developed for many domains such as the health sector both in terms of disease and health services, entertainment fields such as film reviews, music, to the political field.

AIM AND OBJECTIVES OF THE STUDY

The aim of this project is to analyze the sentiments about opay applications in the Google Play Store and to determine the level of user satisfaction based on comments in the Google Play Store review column and to determine the level of service success, shortcomings and how to further improve the app for better customer experience based on community sentiments. The objectives include:

1. Classifying the extracted data using Random forest as a supervised data mining algorithm to predict good and bad reviews.
2. To analyzing the reviews of users on opay app to extract sentiment
3. To get valuable insight on the level of app success, shortcomings and how to further improve the app for better customer experience

RESULT SUMMARY

The trained random forest classification model had an accuracy of 86%, precision of 87%, recall of 86% and average cross validated ROC AUC of 90% indicating a relatively good model. The most important features were the 4 sentiment scores (positive, compound, neutral and negative), word and character counts, a few document vectors, and the words great and easy (from TD-IDF). The model was then applied to the neutral reviews to try and categorize the review into positive or negative.

- The RFC model predicted 564 reviews as good incorrectly and 445 review as bad incorrectly.The PR curve shows the calculated precision and recall at various threshold values. The precision values for our model remain relatively stable at each threshold AP= 0.95
- After extracting sentiment from the reviews using vader module, positive reviews had the highest number of 73% while negative reviews had the lowest number of 27%. According to the number of reviews positive reviews had the highest number of reviews which means people are satisfied with the app but we can’t base the app success on only the positive reviews we have to strike a balance in order to get good insight,
- App success: Result shows that users are happy and like the app because it is fast, efficient and easy to use. Some key functions that users enjoyed were Oride service, electricity bill payment, airtime fill up, TV subscriptions amidst other services.
- Shortcoming: many of users complained of experiencing technical issues of the app not able to connect to the internet on their mobile phones, also geo-location for oride service not accurate causing late arrival of riders , riders misbehave, agents Unable get commission on

In [1]:
```
import warnings
warnings.filterwarnings('always')
warnings.filterwarnings('ignore')
from sklearn.feature_extraction.text import TfidfVectorizer, CountVectorizer
from sklearn.metrics import accuracy_score, classification_report, confusion_m
atrix,roc_curve, auc
from sklearn.metrics import roc_auc_score, average_precision_score, precision_
recall_curve
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split, cross_val_score
# from sklearn.utils.fixes import signature
from nltk.corpus import wordnet, stopwords
from nltk import pos_tag, ngrams
from nltk.tokenize import WhitespaceTokenizer
from nltk.stem import WordNetLemmatizer
from nltk.sentiment.vader import SentimentIntensityAnalyzer
from nltk.collocations import * #N-grams
import nltk.collocations #N-grams
from collections import Counter #N-grams
from gensim.test.utils import common_texts
from gensim.models.doc2vec import Doc2Vec, TaggedDocument
from wordcloud import WordCloud
import string
import seaborn as sns
import matplotlib.pyplot as plt; plt.rcdefaults()
from matplotlib import rc
import missingno as msno #missmap
import numpy as np
import pandas as pd
import os
```

In [50]:
```
#Set home drive
os.chdir("C:/Users/Oluwaseyi/Documents/final year project code/sentiment-analy
sis-on-opay-app/")#set drive
pd.set_option('display.max_colwidth', -1) #set column width for better string
 viewing
```

Exploratory Data Analysis (EDA)

This data was provided by a third party Appfollow. To perform similar analysis, a sample data set obtained from appfollow website.

Data Description

The app store data used here contains 44,490 observations and 26 colums. Each customer review is composed of Date, AppID, AppName,Language,Version, Version Code, Rating, Title, Review, Translated title, Translated review, Reply Date, Developer Reply, User, Device, Device Type,Tags, Categories, Notes, Likes, Dislikes, Link, Permalink, AF Link.

In [51]:
```
#Loading the data and initial preview
df = pd.read_csv("reviews_googleplay_1591435589.csv", encoding = "ISO-8859-1")
#Loading the data
print('Dimensions:',df.shape) #call data dimensions
df.dtypes
df.head()
```

```
Dimensions: (44490, 26)
```

Out[51]:

	Date	AppID	AppName	Language	Version	VersionCode	OS	Author	Rating
0	6/4/2018 6:48	team.opay.pay	OPay - OMall, ORide, Airtime, Transfer & more	en	NaN	NaN	NaN	Ethel Mwanyalo	5
1	6/11/2018 10:35	team.opay.pay	OPay - OMall, ORide, Airtime, Transfer & more	en	NaN	NaN	NaN	Abiola Daramola	5
2	6/13/2018 12:06	team.opay.pay	OPay - OMall, ORide, Airtime, Transfer & more	en	NaN	NaN	NaN	Yahaya Saminu	5
3	6/15/2018 11:28	team.opay.pay	OPay - OMall, ORide, Airtime, Transfer & more	en	NaN	NaN	NaN	A Google user	5
4	6/18/2018 11:40	team.opay.pay	OPay - OMall, ORide, Airtime, Transfer & more	en	NaN	NaN	NaN	Abena Music	5

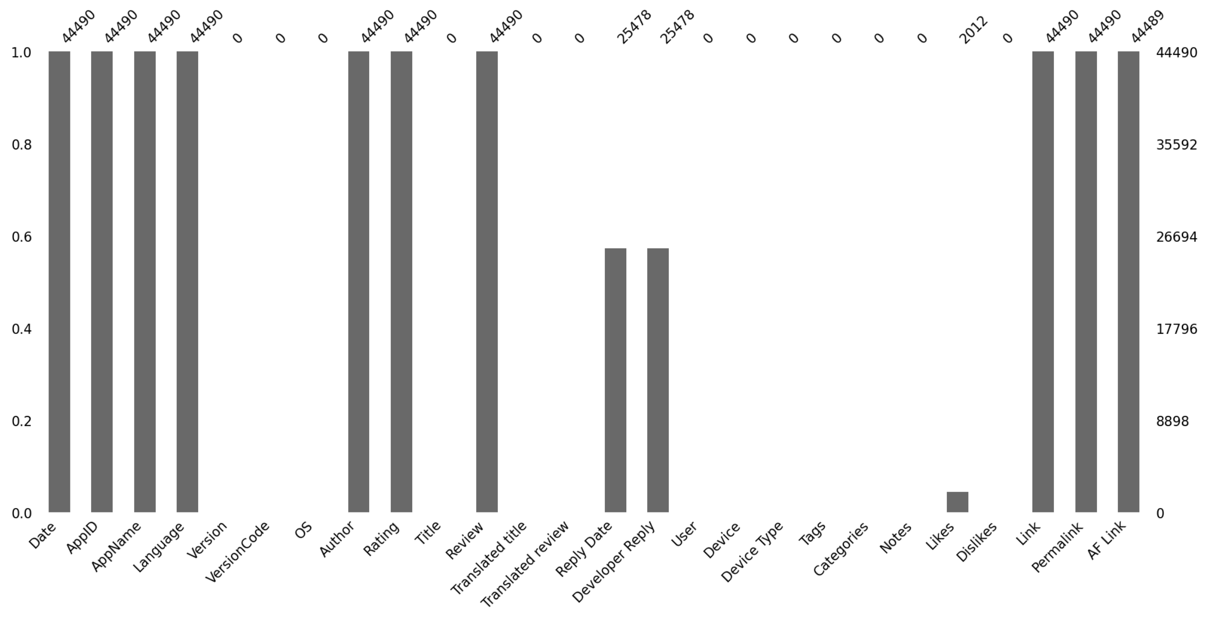
5 rows × 26 columns

Data Integrity

We can see that the fields TranslatedTitle, TranslatedReview, User, Device, DeviceType, Tags, Version, Version Code, os, Title, Categories and Notes have no data. These can be removed during the data cleaning phase. Additionally, ReplyDate, DeveloperReply, Title and Author are quite sparsely populated fields.

```
In [52]: msno.bar(df)
```

```
Out[52]: <matplotlib.axes._subplots.AxesSubplot at 0x45843a88>
```



Initial Data Cleaning

To simplify the data, features that are less meaningful to our analysis or are too scarcely populated are removed from the dataset.

> Author, User – Removed unique identifiers.
> AppName, AppID - is in a cleaner format and more consistent. but of no meaning to the analysis
> .
> Language – The reviews should all be in English so this field is not meaningful
> TranslatedTitle, TranslatedReview, User, Device, DeviceType, ReplyDate, DeveloperReply, Tags, Notes,Version, Version Code, os, Title, Categories – too scarcely populated or have no data to provide any meaningful insight.
> Link – Removed as outside of the scope of this project, maybe a field of interest for further analysis.
> .

In [53]:
```
df = df.drop(["Author","AppName","Version","VersionCode","OS","Language","Tran
slated title", "Translated review", "Reply Date","Developer Reply","User","Dev
ice","Device Type","Tags","Notes","Link","Likes","Dislikes","Permalink","AF Li
nk","Categories","Title"], axis=1)
df['AppID']=df['AppID'].replace({'team.opay.pay': 'Opay'})
print('Dimensions:',df.shape)
df.head()
```

```
Dimensions: (44490, 4)
```

Out[53]:

	Date	AppID	Rating	Review
0	6/4/2018 6:48	Opay	5	Great stuff loading
1	6/11/2018 10:35	Opay	5	Good
2	6/13/2018 12:06	Opay	5	Nice
3	6/15/2018 11:28	Opay	5	Cool app
4	6/18/2018 11:40	Opay	5	How do it work to earn..pls urgently..

In [54]:
```
#Total number rows and columns
df.shape
```

Out[54]:
```
(44490, 4)
```

In [55]:
```
# remove duplicates/ for every duplicate we will keep only one row of that typ
e.
df.drop_duplicates(subset=['Rating','Review'],keep='first',inplace=True)
print(df.shape)
df.head()
```

```
(26535, 4)
```

Out[55]:

	Date	AppID	Rating	Review
0	6/4/2018 6:48	Opay	5	Great stuff loading
1	6/11/2018 10:35	Opay	5	Good
2	6/13/2018 12:06	Opay	5	Nice
3	6/15/2018 11:28	Opay	5	Cool app
4	6/18/2018 11:40	Opay	5	How do it work to earn..pls urgently..

In [56]:
```
df["Reviews"]=df["Review"]
df.drop(['Review'],axis=1,inplace=True)
df.head()
```

Out[56]:

	Date	AppID	Rating	Reviews
0	6/4/2018 6:48	Opay	5	Great stuff loading
1	6/11/2018 10:35	Opay	5	Good
2	6/13/2018 12:06	Opay	5	Nice
3	6/15/2018 11:28	Opay	5	Cool app
4	6/18/2018 11:40	Opay	5	How do it work to earn..pls urgently..

DATA PREPARATION AND CLEANING

My initial attempts at lemmatizing the review text were unsuccessful as a spot check of the corpus showed many words that were not transformed to their base form. Upon further research, it was noted that the default setting for the lemmatization module in NTLK wordnet was ‘noun’ resulting in the transformation of only noun words. To resolve this, the function below defines the word type based on the position tag obtained from the NLTK pos_tag module (the pos_tag module is applied in the clean_text function in the following section)

In [57]:
```
def get_tag(pos_tag):
    if pos_tag.startswith('J'):
        return wordnet.ADJ
    elif pos_tag.startswith('V'):
        return wordnet.VERB
    elif pos_tag.startswith('N'):
        return wordnet.NOUN
    elif pos_tag.startswith('R'):
        return wordnet.ADV
    else:
        return wordnet.NOUN
```

The clean_text function defined below applies the following transformations:

> 1) Change all words to lower case (lemmatization does not work on capitals as they are assumed to be proper nouns).
> 2) Tokenize the text and remove punctuation.
> 3) Remove numeric values.
> 4) Remove stop words (using pre-built stop word dictionary).
> 5) Remove any empty tokens.
> 6) Apply a position tag to each word and define it based on the previously defined get_tag function as adjective, noun, verb, or adverb.
> 7) Lemmatize the words.
> 8) Remove any single letter words resulting from lemmatization.

In [58]:
```
import re
def clean_text(text):
    text = text.lower() #change all text to lower case
    text = re.sub(r'\s*(?:https?:\/\/)?[\w.-]+(?:\.[\w.-]+)+[\w\-._~:/?#[\]@!
\$&\'\(\)\*\+,;=.]+','',text)
    text = [word.strip(string.punctuation) for word in text.split(" ")] #token
ize and remove punctuation
    text = [word for word in text if not any(c.isdigit() for c in word)] #remo
ve numeric values
    stop = stopwords.words('english') #call english stop word dictionary
    text = [x for x in text if x not in stop]#remove stop words
    text = [t for t in text if len(t) > 0] #remove empty tokens
    pos_tags = pos_tag(text)#apply position tag to text
    text = [WordNetLemmatizer().lemmatize(t[0], get_tag(t[1])) for t in pos_ta
gs] #apply pos_tag function and lemmatize text
    text = [t for t in text if len(t) > 1]# remove single letter words
    text = " ".join(text) #combine
    return(text)
#create new column with cleaned text
df["reviews_clean"] = df["Reviews"].apply(lambda x: clean_text(x))
```

In [59]:
```
#Text Before Text Cleaning
print('Before Text Cleaning')
df['Reviews'].head()
```

```
Before Text Cleaning
```

Out[59]:
```
0    Great stuff loading
1    Good
2    Nice
3    Cool app
4    How do it work to earn..pls urgently..
Name: Reviews, dtype: object
```

In [60]:
```
#Text Before Text Cleaning
print('After Text Cleaning')
df['reviews_clean'].head()
```

```
After Text Cleaning
```

Out[60]:
```
0    great stuff loading
1    good
2    nice
3    cool app
4    work urgently
Name: reviews_clean, dtype: object
```

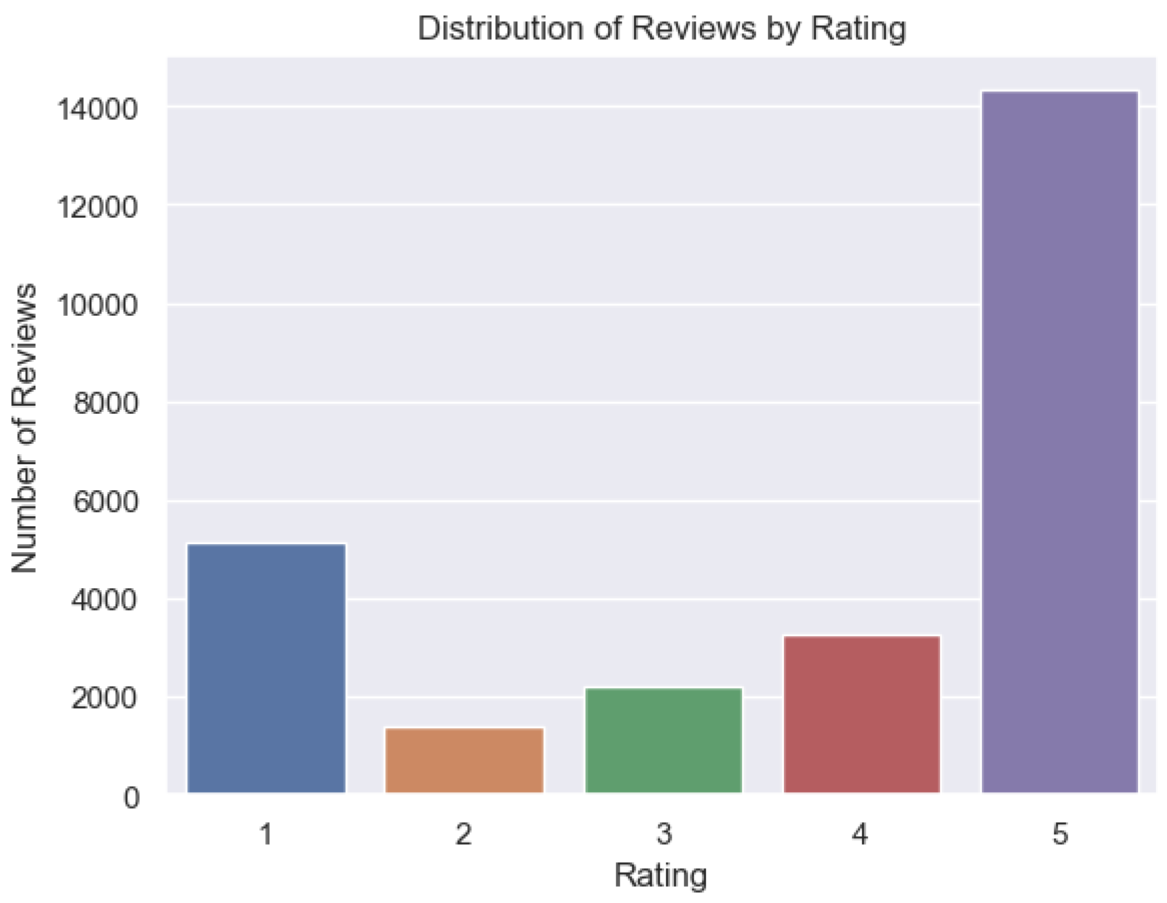
In [61]:
```
# Drop all columns that are blank as a results of the text cleaning function.
 Lost 176 rows.
print(df.shape)
df = df[df['reviews_clean'].map(len) > 0]
print(df.shape)
```

```
(26535, 5)
(26359, 5)
```

Plot: Distribution of Reviews by Rating

This plot views the distributions of reviews across all ratings. We can see that the number of positive reviews has the highest number of reviews, causing our dataset to be imbalanced, followed by negative reviews having less than 6000 comments.

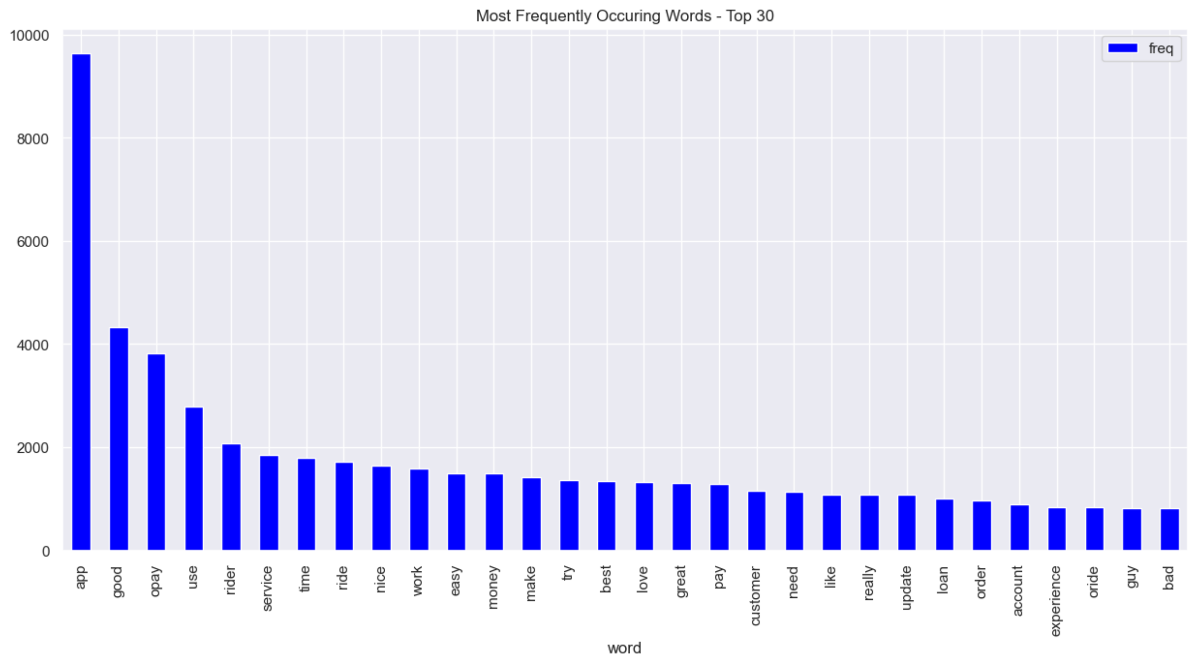
In [62]:
```
sns.set(style="darkgrid")
bx = sns.countplot(x = "Rating", data=df)
bx.set(xlabel='Rating', ylabel='Number of Reviews',title='Distribution of Revi
ews by Rating')
plt.show()
```



Most Frequently Occuring Words

In [63]:
```
from sklearn.feature_extraction.text import CountVectorizer

cv = CountVectorizer(stop_words = 'english')
words= cv.fit_transform(df['reviews_clean'].values.astype('U'))

sum_words = words.sum(axis=0)

words_freq = [(word, sum_words[0, i]) for word, i in cv.vocabulary_.items()]
words_freq = sorted(words_freq, key = lambda x: x[1], reverse = True)

frequency = pd.DataFrame(words_freq, columns=['word', 'freq'])

frequency.head(30).plot(x='word', y='freq', kind='bar', figsize=(15, 7), color
= 'blue')
plt.title("Most Frequently Occuring Words - Top 30")
```

Out[63]:
```
Text(0.5, 1.0, 'Most Frequently Occuring Words - Top 30')
```



In [64]:
```
dfn= df[df['reviews_clean'].str.contains("app")]
#dfn_pg['Reviews'].head(10)
dfn['reviews_clean'].head(10)
```

Out[64]:
```
3     cool app
6     apply loan
11    useless ewallet balance even reflect app can't buy \nairtime pay bill
i'm fear money pump name \nof top ewallet
14    least let get update work tire update app \ndoesn't get transaction exe
cute
16    think best app e-wallet transfer work perfectly \nin mine
32    best app sow far please need customer phone number
35    good app credit debit also use buy credit data
36    can't send money bank account use app top account cant send money bad
40    good app user friendly
43    best money app
Name: reviews_clean, dtype: object
```

FEATURE ENGINEERING

Sentiment Analysis

The Vader module from NLTK was the model selected for sentiment analysis. The Vader module uses a prebuilt lexicon of words to calculate a sentiment score. This module was selected for sentiment analysis because the module takes into consideration the context of the text. The module returns 4 values: positivity score, neutrality score, negativity score and summary score.

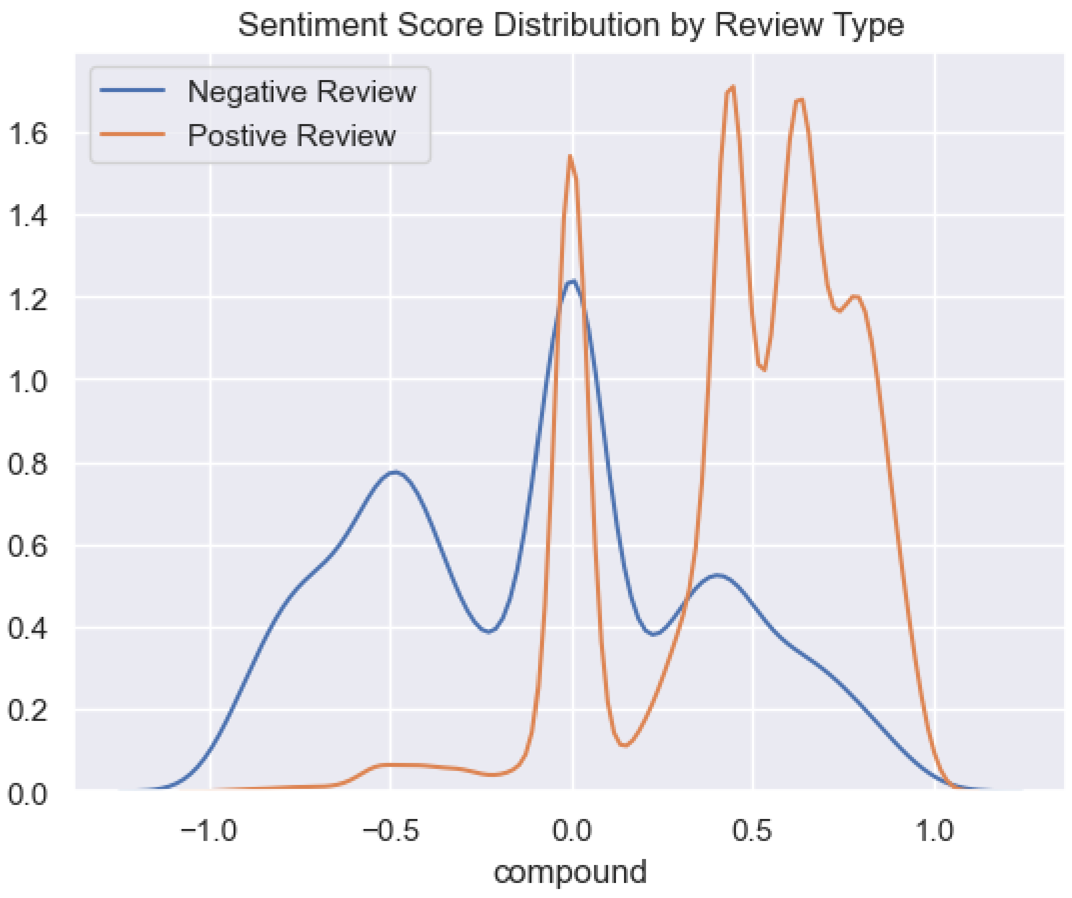
In [65]:
```
sid = SentimentIntensityAnalyzer()
#calculates the negativity,  neutrality, positivity and overall sentiment scor
es
df["sentiments"] = df["reviews_clean"].apply(lambda x: sid.polarity_scores(x))
#drop sentiment column and add the 4 sentiment scores as separate features to
 primary dataset
df = pd.concat([df.drop(['sentiments'], axis=1), df['sentiments'].apply(pd.Ser
ies)], axis=1)
df[['AppID','Rating','reviews_clean','neg','neu','pos','compound']].head(10)
```

Out[65]:

	AppID	Rating	reviews_clean	neg	neu	pos	compound
0	Opay	5	great stuff loading	0.0	0.328	0.672	0.6249
1	Opay	5	good	0.0	0.000	1.000	0.4404
2	Opay	5	nice	0.0	0.000	1.000	0.4215
3	Opay	5	cool app	0.0	0.303	0.697	0.3182
4	Opay	5	work urgently	0.0	1.000	0.000	0.0000
5	Opay	2	unable either send receive cash seem like fully \nsupported	0.0	0.579	0.421	0.6240
6	Opay	5	apply loan	0.0	1.000	0.000	0.0000
7	Opay	1	can't get access code verify phone number	0.0	0.822	0.178	0.0772
8	Opay	4	love	0.0	0.000	1.000	0.6369
9	Opay	4	can't find option put amount want recharge	0.0	0.822	0.178	0.0772

The graph below shows the compound sentiment calculated by Vader distributed by good and bad reviews. We can see that good reviews are mostly considered very positive by Vader, whereas, bad reviews are more dispersed with a slightly higher proportion of negative reviews with negative sentiment scores. The only variation to this trend is the slight peak around the neutral compound score (zero) for both negative and positive reviews.

In [66]:
```
for x in [1, 5]:
    subset = df[df['Rating'] == x]
    if x > 3:
        label = "Postive Review"
    else:
        label = "Negative Review"
    sns.distplot(subset['compound'], hist = False, label = label).set_title('S
entiment Score Distribution by Review Type')
```

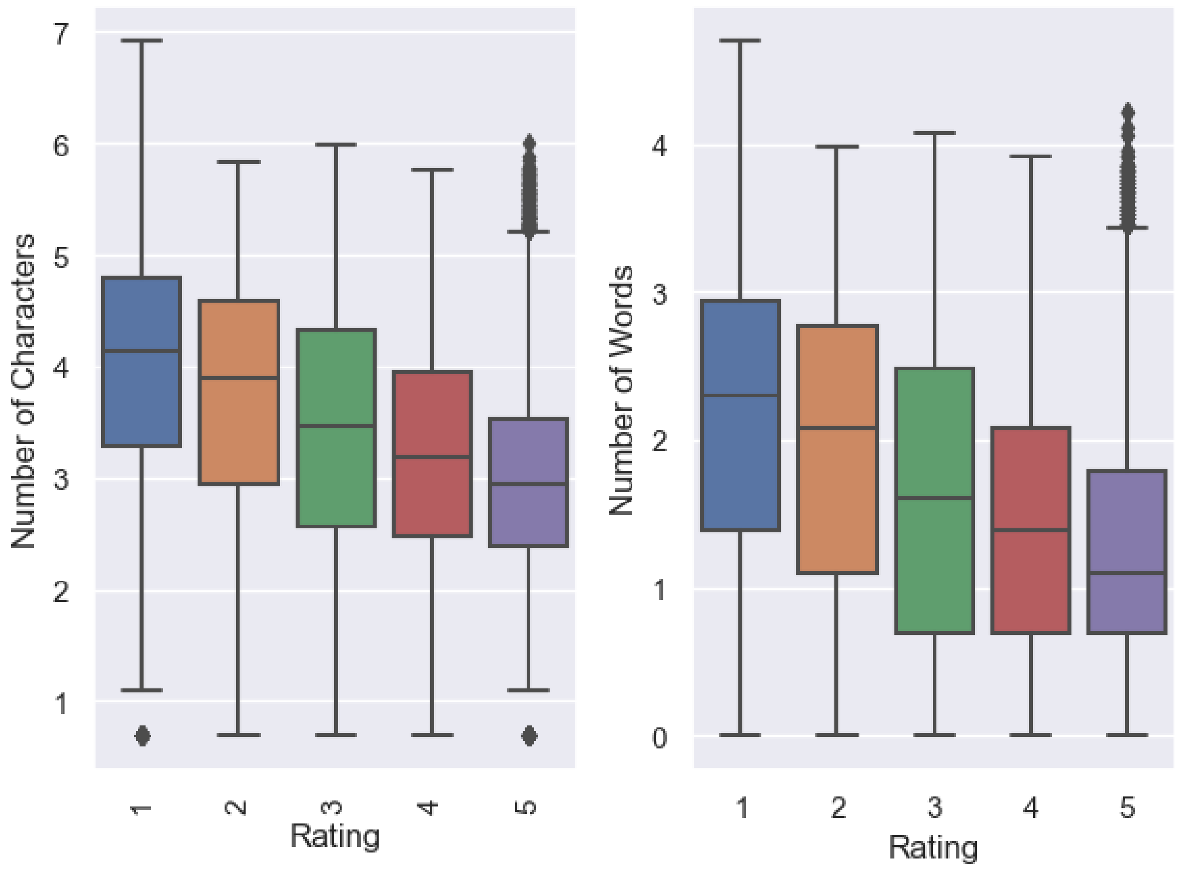


Word and Character Count Features

Two new features are created by extracting the number of characters and number of words per review. Log transformation is applied to pull in outliers.

In [67]:
```
df["num_chars"] = df["reviews_clean"].apply(lambda x: len(x))
df["num_words"] = df["reviews_clean"].apply(lambda x: len(x.split(" ")))
#log transformation
df['num_chars1'] = np.log(df['num_chars'])
df['num_words1'] = np.log(df['num_words'])
```

In [68]:
```
x1 = df['Rating']
x2 = df['Rating']
y1 = df['num_chars1']
y2 = df['num_words1']
#plot num_chars by rating in column 1
plt.subplot(1, 2, 1)
plt.xticks(rotation=90)
g = sns.boxplot(x1, y1)
g.set(xlabel='Rating', ylabel='Number of Characters',title='')
#plot num_words by rating in column 2
plt.subplot(1, 2, 2)
g = sns.boxplot(x2, y2)
g.set(xlabel='Rating', ylabel='Number of Words',title='')

plt.tight_layout()
plt.show()
```



We can see a trend forming from the boxplot below, where users tend to leave longer reviews for negative ratings (<3) and neutral ratings (=3) and shorter reviews good review (>3). This may be a useful feature for our predictive models

Doc2Vec Feature Creation

The doc2vec method from the Genism module is used to generate document vectors for each cleaned review. The doc2vec module uses a modified word2vec model with the addition of a document unique vector, which numerically represents the document. This provides a document-concept representation of each review. This feature is important for training our model since similar texts should have similar vector representations. We first start by creating doc2vec vector columns and then proceed to train the model. The model is then applied to the text to transform each review into vector data before being combined with our original dataframe.

Warning Message to install compiler to speed up genism is not necessary for the size of data used in this notebook. For larger data, a compiler would be recommended as this model took roughly 26 minutes to run.

In [69]:
```
documents = [TaggedDocument(doc, [i]) for i, doc in enumerate(df["reviews_clea
n"].apply(lambda x: x.split(" ")))]
# train a Doc2Vec model with our text data
model = Doc2Vec(documents, vector_size=6, window=2, min_count=1, workers=4)
# transform each document into a vector data
df_vector = df["reviews_clean"].apply(lambda x: model.infer_vector(x.split(" "
))).apply(pd.Series)
df_vector.columns = ["df_vector_" + str(x) for x in df_vector.columns]
df = pd.concat([df, df_vector], axis=1)
```

Term Frequency - Inverse Document Frequency

The word frequency is calculated using the TF-IDF model. In addition to just counting word frequency, this model computes the relative importance of each word based on the frequency of occurrence of the word in each text. A column is generated for every word which occurs in a minimum of 10 different documents to provide a relative filter on importance and to remove size. This can be adjusted to fine tune the predictive models.

In [70]:
```
# add tf-idfs columns
from sklearn.feature_extraction.text import TfidfVectorizer
tfidf = TfidfVectorizer(min_df = 10)
tfidf_result = tfidf.fit_transform(df["reviews_clean"]).toarray()
tfidf_df = pd.DataFrame(tfidf_result, columns = tfidf.get_feature_names())
tfidf_df.columns = ["word_" + str(x) for x in tfidf_df.columns]
tfidf_df.index = df.index
df = pd.concat([df, tfidf_df], axis=1)
```

Define Good and Bad Reviews

The final feature created is to define a bad review (rating < 3) by denoting it with 0 and all other ratings with 1. For the purposes of our model, the neutral reviews (rating of 3) are separated into another dataframe. Our dataset is relatively imbalanced with 73% good review and 27.0% bad reviews oversampling of our dataset would take care of the imbalanced to make it a balanced dataset.

In [71]:
```
df.shape
```
Out[71]:
```
(26359, 1562)
```

In [72]:
```
df['label'] = np.where(df['Rating']<3, 0, 1)
#take lowest and highest rating
df_class = df[(df['Rating'] < 3) | (df['Rating'] > 3)]
df_neutral = df[(df['Rating'] == 3)].drop(['label'], axis=1)
df_class = df_class.sort_values(by=['Rating'])
print ("Dimenions:", df_class.shape)
print ("Good (1) vs Bad (0) split:" "\n",df_class["label"].value_counts(normal
ize = True))
df_class.groupby('label').count()
```

```
Dimenions: (24142, 1563)
Good (1) vs Bad (0) split:
 1    0.729268
0    0.270732
Name: label, dtype: float64
```

Out[72]:

	Date	AppID	Rating	Reviews	reviews_clean	neg	neu	pos	compound	num_ch
label										
0	6536	6536	6536	6536	6536	6536	6536	6536	6536	65
1	17606	17606	17606	17606	17606	17606	17606	17606	17606	176

2 rows × 1562 columns

MODEL DEVELOPMENT

The Random Forest model (RF) is used to predict if a review is good or bad given the various features we created from the review text. The model will then be used on the neutral dataset (rating = 3) to categorize the reviews.

Random Forest Classifier

The features used to train the RF model are selected and any columns to be ignored are defined. The dataset is then split into training and test datasets.

In [73]:
```
# feature selection
label = "label"
ignore_cols = [label, "Reviews", "reviews_clean", "Date", "AppID","Rating"]
features = [c for c in df_class.columns if c not in ignore_cols]
# split the data into train and test
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(df_class[features], df_cla
ss[label], test_size = 0.3, random_state = 42)
```

The resultant dataset for training is 16,899 rows x 1,558 columns and the test dataset is 7,243 rows x 1,558 columns. The 80/30 split was used as the app dataset is relatively small.

In [74]:
```
print('Training Features Shape:', X_train.shape)
print('Training Labels Shape:', y_train.shape)
print('Testing Features Shape:', X_test.shape)
print('Testing Labels Shape:', y_test.shape)
```

```
Training Features Shape: (16899, 1557)
Training Labels Shape: (16899,)
Testing Features Shape: (7243, 1557)
Testing Labels Shape: (7243,)
```

Balancing our label class

In [75]:
```
from imblearn.combine import SMOTETomek
```

In [76]:
```
os=SMOTETomek(1)
X_train_ns,y_train_ns=os.fit_sample(X_train,y_train)
print("The number of classes before fit {}".format(Counter(y_train)))
print("The number of classes after fit {}".format(Counter(y_train_ns)))
```

```
The number of classes before fit Counter({1: 12319, 0: 4580})
The number of classes after fit Counter({1: 12157, 0: 12157})
```

imblearn is used to balance our label class using over sampling method so our model would not be biased when prediting and it would be able to predict well on new dataset. Our inital label class was 0: 12319, 1: 4580 after imblearn was used we had 0: 12312, 1: 12312 ratio 50:50 making it a balanced dataset.

The RF model is trained and cross validation is run to get a better overview of our model's performance..

In [77]:
```
# train a random forest classifier
rf = RandomForestClassifier(n_estimators = 150, random_state = 101)
rf.fit(X_train_ns,y_train_ns)
#Cross Validation Score
rfc_cv_score = cross_val_score(rf, df_class[features], df_class[label], cv=10,
scoring= 'roc_auc')
```

we use pickle libiary to save our model to avoid retraining everytime we work on our dataset

In [97]:
```
import pickle
with open('random_forest_classifier2.pickle','wb') as f:#saving our model
    pickle.dump(rf,f)

# pickle_in= open('random_forest_classifier1.pickle','rb')# opening our model
# clf= pickle.load(pickle_in)
```

Model Evaluation

Confusion Matrix:

The RFC model predicted 564 reviews as good incorrectly and 445 review as bad incorrectly.

Classification Report:

The model achieved an average precision of 0.86, average recall of 0.86 and average accuracy of 0.86. We can see the model has higher precision when it comes to predicting positive review. This may be because the positive sentiment is one of most important feature for our model (refer to Feature Importance section).

Cross Validated (CV) AUC Score:

The model achieved an average CV AUC score of 0.90 which indicates a relatively good model.

In [79]:
```
print('CONFUSION MATRIX')
print(confusion_matrix(y_test, rf.predict(X_test)))
print('\n')
print (pd.crosstab(y_test, rf.predict(X_test), rownames=['Actual Result'], col
names=['Predicted Result']))
print('\n')
print('CLASSIFICATION REPORT')
print(classification_report(y_test, rf.predict(X_test)))
print('\n')
print('ALL AUC SCORES')
print(rfc_cv_score)
print('\n')
print('MEAN AUC SCORE: ', rfc_cv_score.mean())
```

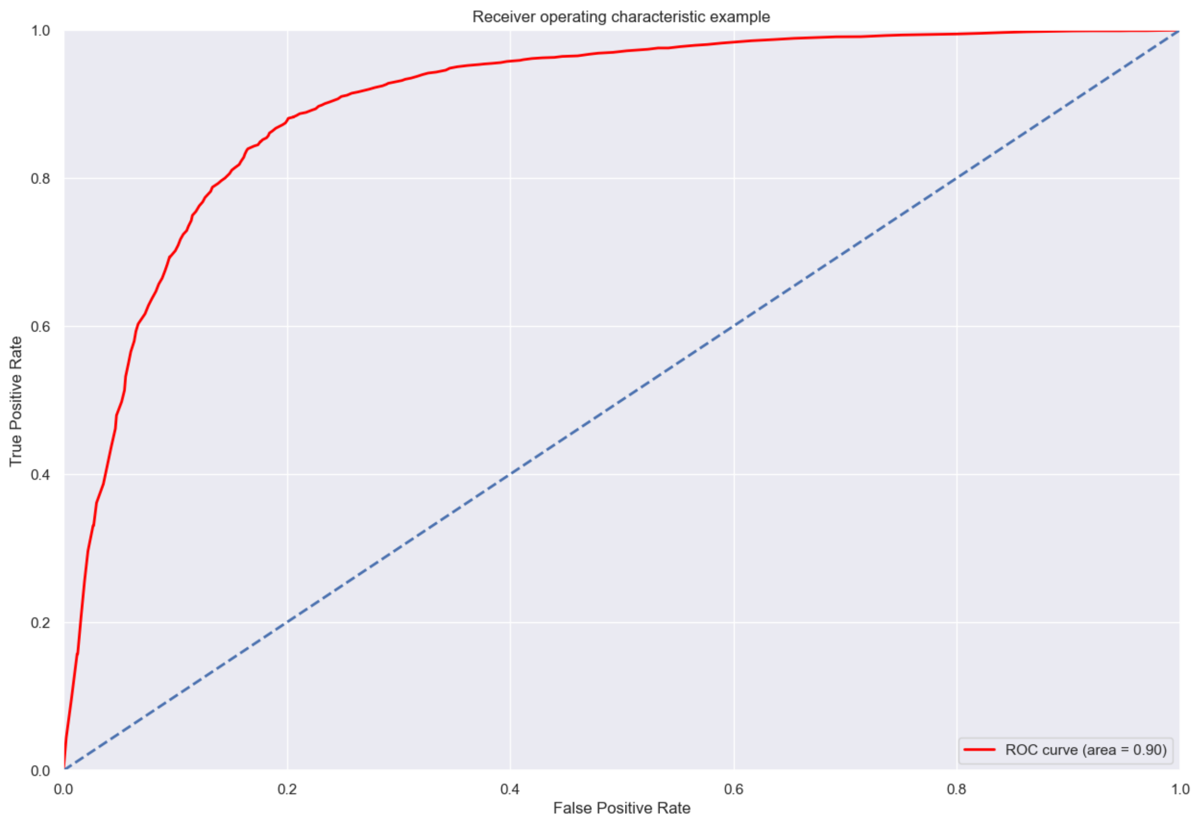
```
CONFUSION MATRIX
[[1509  447]
 [ 544 4743]]


Predicted Result     0     1
Actual Result
0                 1509   447
1                  544   4743


CLASSIFICATION REPORT
              precision    recall  f1-score   support

           0       0.74      0.77      0.75      1956
           1       0.91      0.90      0.91      5287

    accuracy                           0.86      7243
   macro avg       0.82      0.83      0.83      7243
weighted avg       0.87      0.86      0.86      7243



ALL AUC SCORES
[0.86523547 0.89011491 0.90828208 0.90139439 0.91017906 0.93943955
 0.93123252 0.91447937 0.8901036  0.87808333]


MEAN AUC SCORE:  0.9028544261567095
```

Receiver Operating Characteristics (ROC) Curve

The trade-off between the true positive (TP) and false positive (FP) rate is shown in the Receiver Operating Characteristics (ROC) curve, and can be used to access the quality of the classifier used in our model. The distance between the ROC curve and the diagonal baseline indicates the reliability of the predictions from our model. The model is quite good with an area under curve (AUC) value of 0.91.

Note: ROC is not a good indicator of model quality if the data is skewed towards a specific outcome as this could mute the FP and FN prediction rates (depending on the skewing of data). The app data review was relatively balanced in terms of the number of defined good or bad reviews, which give us some confidence in the ROC curve

In [80]:
```
y_pred = [x[1] for x in rf.predict_proba(X_test)]
fpr, tpr, thresholds = roc_curve(y_test, y_pred, pos_label = 1)
roc_auc = auc(fpr, tpr)
plt.figure(1, figsize = (15, 10))
lw = 2
plt.plot(fpr, tpr, color='red',
         lw=lw, label='ROC curve (area = %0.2f)' % roc_auc)
plt.plot([0, 1], [0, 1], lw=lw, linestyle='--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.0])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver operating characteristic example')
plt.legend(loc="lower right")
plt.show()
```



Precision Recall Curve (Average Precision)

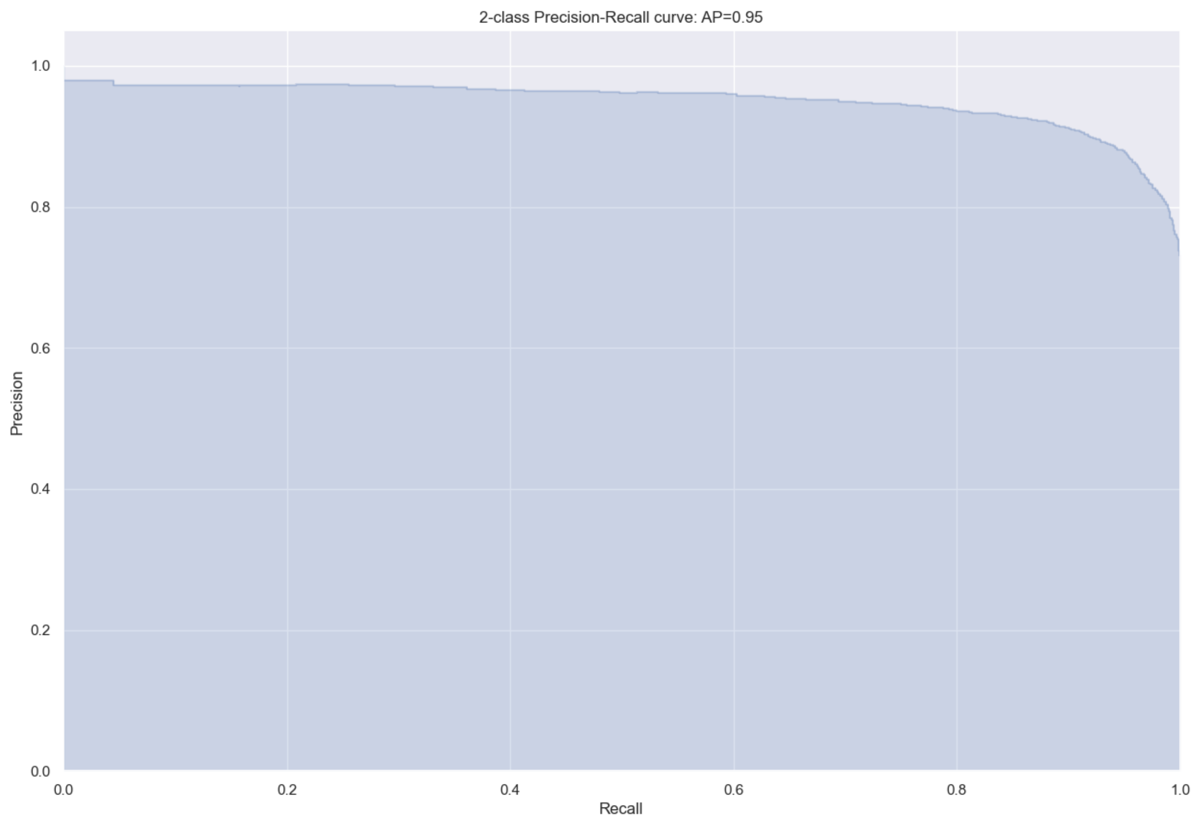
The PR curve shows the calculated precision and recall at various threshold values. The precision values for our model remain relatively stable at each threshold AP= 0.95.

> Precision (positive prediction value) is the ratio of TP/ (TP + FP)
> Recall (sensitivity) is the ratio of TP/(TP + FN)

Note: The PR curve is useful for dataset that are imbalanced.

In [81]:
```
from funcsigs import signature
average_precision = average_precision_score(y_test, y_pred)
precision, recall, _ = precision_recall_curve(y_test, y_pred)
step_kwargs = ({'step': 'post'}
               if 'step' in signature(plt.fill_between).parameters
               else {})

plt.figure(1, figsize = (15, 10))
plt.step(recall, precision, color='b', alpha=0.2,
         where='post')
plt.fill_between(recall, precision, alpha=0.2, color='b', **step_kwargs)

plt.xlabel('Recall')
plt.ylabel('Precision')
plt.ylim([0.0, 1.05])
plt.xlim([0.0, 1.0])
plt.title('2-class Precision-Recall curve: AP={0:0.2f}'.format(average_precisi
on))
```

Out[81]:
```
Text(0.5, 1.0, '2-class Precision-Recall curve: AP=0.95')
```



Feature Importance

The most important features were the 4 sentiment scores generated by Vader,the doc2vec, the number of words and characters features. Additionally, some words identified by TF-IDF such as “app” and "good" have fairly high importance, also may be correlated with the Vader “pos” scores and the words identified by TF-IDF such as "bad" may be correlated with the Vader “neg” scores, while the words dentified by TF-IDF such as "update","download", "can" may be correlated with the Vader “neu” scores.

In [ ]:

In [82]:
```
# show feature importance
feature_importances_df = pd.DataFrame({"feature": features, "importance": rf.f
eature_importances_}).sort_values("importance", ascending = False)
feature_importances_df.head(20)
```

Out[82]:

	feature	importance
2	pos	0.076610
3	compound	0.070698
0	neg	0.044265
1	neu	0.041633
9	df_vector_1	0.033498
7	num_words1	0.031259
5	num_words	0.030501
6	num_chars1	0.029769
8	df_vector_0	0.029394
12	df_vector_4	0.027643
4	num_chars	0.026245
10	df_vector_2	0.024661
13	df_vector_5	0.023368
11	df_vector_3	0.022032
103	word_app	0.014389
588	word_good	0.010922
149	word_bad	0.009752
1451	word_update	0.009485
399	word_download	0.009431
214	word_can	0.009093

Model Application

The RF model is applied to the dataset with ratings of 3 to determine if the reviews are good or bad.

```
In [83]: df_temp = df_neutral[['Date','AppID','Rating','Reviews','reviews_clean']]
         df_neutral = df_neutral.drop(['Reviews','reviews_clean','Date','AppID','Ratin
         g'], axis=1)
         df_neutral['prediction'] = rf.predict(df_neutral)
         df_neutral = pd.concat([df_temp.reset_index(drop=True),df_neutral.reset_index(
         drop=True)], axis=1)
```

Preview Predicted Reviews

Predicted Good Reviews

- The predicted good review preview seems to be less insightful, but this is somewhat expected given the high neutrality score noted during the Vader sentiment step. We see the model has failed to identify sarcasm in line 10: “Pay first before we rate the app”, “please how can i upgrade my account? i want to fund more than N10,000 to my account” and “The app is great. But please trying including an option for transaction cancelation. And the airtime limit of 200 is much, the data too please add daily subscription too …”, "want to be an agent how can I go about it ".
- In line 23:“The app is great. But please trying including an option for transaction cancelation. And the airtime limit of 200 is much, the data too please add daily subscription too …” the app lacks this features rating may improve if in further update this feature are added to the app

We noted that our model weighed positive words heavily in feature importance, which would explain these results.

In [84]:
```
print('Predicted Good Reviews')
dfn_pred_good = df_neutral[(df_neutral['prediction'] == 1)]
dfn_pred_good['Reviews'].head(20)
```

```
Predicted Good Reviews
```

Out[84]:
```
0     Good
2     Can I use these OPay and send money to Nigeria?
3     Its ik
6     Wonderful
7     Opay transfer charges are lesser than what banks charge,, and also the
fact that your pos transaction are processed same day is good. however your c
harge on transaction (0.99%) is quite high, also the number of weeks one has
to wait before he gets a machine should be treated. Again 6million per month
on transactions with machine is somehow high IMO.
8     It's been good so far but needs more perfection
9     good
10    perfect
12    Good App I like it, transactions that I've done on the app are just ins
tant. keep the good work and keep improving. I like to know how I can pay my
electricity bill on the app please can any of your amiable CR put me through
on that. thanks much.
15    Though the app is efficient and fast but so many features such as airti
me,transaction receipt printout,provision for saving an account as a benefici
ary,other betting companies,tv subcriptions etc. Thanks
16    please how can i upgrade my account? i want to fund more than N10,000 t
o my account
17    it's very good, there is room for improvement
18    best services
19    Nice to use
20    ok
22    want to be an agent how can I go about it
23    The app is great. But please trying including an option for transaction
cancelation. And the airtime limit of 200 is much, the data too please add da
ily subscription too
24    Mordred
27    fair
29    doing good
Name: Reviews, dtype: object
```

Predicted Bad Reviews

For the most part, the model seems to have done a pretty good job of categorizing the neutral reviews. Based on the preview we can see a couple of issues being highlighted by reviewers: : Alert notification issues : Sender unable to receive alert from Opay as there receipt

- Lack of services: use of debit card, POS you uable to receive money from other banks
- Error in services: Referral not working, unable to login, takes time to connect to the internet (app is slow), app unable to connect to mobile network, geolocation service not accurate
- Errors in transactions: failed EFT, failed deposit, etc
- Opay agents issues: Unable get commission on some services, charges on transcation is high
- Oride service issues: late arrival time of divers, divers are unmannered, divers reject long distance trip, drivers charge off the book

In [85]:
```
print('Predicted bad Reviews')
dfn_pred_bad = df_neutral[(df_neutral['prediction'] == 0)]
dfn_pred_bad['Reviews'].head(50)
```

```
Predicted bad Reviews
```

Out[85]:
```
1      Pay first before we rate the app
4      We want d sender to be receiving alert from Opay as there receipt dat
d money have getting to d receiver account
5      The app is OK but my account was blocked and I have a lot of money in
there
11     Pls i deposited some certain amount of money to my opay account from m
y normal bank account it was deducted but opay showing my transaction failed
why and no money deposited in my opay account
13     i was rub and my phone and sim was stolen how can i get access to my a
ccount again cus i have money inside
14     is ok
21     Good app,but u need to give agents commission on some payments. Exampl
es are electricity, data, cable TV etc. with that,we will be able to make mon
ey on the platform.
25     Hi, I took a ride with ORide on the 30th of May 2019 from Allen, Ikeja
to Mile 12, however, it appear that the ride was not ended by the rider and I
was eventually charged over â¦11000 after the rider driver end the ride the
following day, please help fix it.
26     the app is very nice...but they should provide ATM card for withdrawer
instead of transfering the money to bank account before you can withdraw mone
y
28     ls good to know who cslled u becos some people can be stupid enough wh
en u ask who is on the line they cut it off
31     Refuse to open
33     firsr ride was ok after i downloaded the app yesterday ,but since then
if i requst for ride i wait for hours and no bike shows up,.now the app is ju
st telling me network failure ,..so not cool
34     i just set up my account and its already telling me that my account is
logged in on another phone.
35     Oride is frustrating. It doesn't give request to nearby riders, but lo
ng distance riders. Work on it. Thank you.
37     this service would have been great, except drivers now prefer to carry
passengers off the app, rather than on the app. During rush hour in the eveni
ng, you would see them line up in ikeja to pick up passengers who are not usi
ng the app.
38     Urgh, I have to sign in every time, well that's not bad since it's als
o a payment app, so security is paramount. I'm rating the app 3 cos I don't g
et ride on time and I've been deducted double for one ride. Beyond this, the
app is good
41     the app is working successful before but now it keep saying Something
went wrong, check your internet connection, and my connection is good, and i
transfer some amount and later it was refunded back to my bank account.
44     This app is wasting of time and was of ðµ
46     all the verification sent didnt get to my mail, instead i was blocked
with my money inside...also the app geolocation needs to be properly checked,
it doesn't send request to the nearest rider;rathet it sends it far away
47     please tell your riders to always pick customer request no matter the
distance... and you people should make sure their papers are complete... beca
use they keep saying they refuse some request because of the incomplete paper
s to avoid being arrested... and harrased... your promoters are the best
51     Bad App
54     Opay came to makurdi but refused to give me pos of which i created acc
ount with them
55     ORide Is not a scam, Am having issue with pay out to the rider man tha
t get me to my destination and Apart from maybe they are trying to resolve th
e issue. Atleast yesterday i was able to recharge through it and my account g
ot credited. So they should try to resolve the issue before i can give them 5
```

```
star thanks
56     not downloading
58     this app no longer open in my phone, every time I try to open it they
say network error,but all other network using apps are working,why?,pls do so
mething abt it
59     a bit delay on a request of a rider to pick u up at d locaton point
60     the app is nice and fast. but i made a transaction which failed and th
ey're yet to reverse and refund my money. I've sent emails to them yet they'v
e not reversed my money.
67     While the app is a very good development in terms of mobile money tech
nology, but keep your agent for a particular period before entrusting them wi
th the POS machine to me I not encouraging after asking for different documen
t still that trust is not there, secondly without the POS you can't receive m
oney from all the banks in Nigeria so how do you want your agent to coup and
thirdly state mostly from the Northern part and North central can't top-up th
ere electricity bills from this App i.e those using kaduna electricity compan
y and PHCN. I'm registered as an agent no utility material, POS and am based
in Kebbi State Nigeria.
71     Dear Opay, my App has refused to connect to internet for sometime now
even though I have data unless connected to wifi, I have updated it, Uninstal
led and re installed still the same, I hope you can help resolve this, and mo
stly riders are not available
72     I think you should work more on making your app user friendly. Then al
so work on high level professionalism from your drivers, some of them dont kn
ow how to talk to customers by insulting them and they always reject peoples
order or turn off their app and be looking for offline customers.
73     your referral system is not working and it took an hour 30 minutes plu
s to see a rider in ibadan
75     it good
76     i am just using it for the first time, made a payment through the mobi
le app it hasnt shown on my okash
77     why is the app not connecting to my internet even when i have a strong
network
78     I have been using th App for days now without any problem. but today t
his App has been asking me to verify my phone number. even after inputting th
OTP, I still can't login.
80     I am unable to access the app despite having enough internet data. Thi
s is unfortunate as I already made a deposit of 1,000 naira into the app for
ORide. Pls admin what is happening?
81     I tried again and this time it was different. So 3 stars i thought you
were having a promo?
82     It's difficult to get a ride. Even when drivers are in cluster close t
o me. Yet I can't still get a ride
83     This application can be sometimes frustrating in search for riders
84     The service is good though but you guys really need to work on our ord
ers. It's annoying to see lots of riders and none of them would get our reque
sts until someone afar picks and arrive late.
87     hard to get a bike
89     i dont really knw wot is happening wit dis app, i have being trying to
download d app buh it was not goin thru
90     you cannot request for bikes even wjen they are right in front of you,
they cannot start trips on their own and are difficult to assign most times
91     The site is telling me that the application will not be downloaded and
installed on my phone Samsung galaxy tab sm-T561.
93     App refuses to connect on mobile connection except WiFi even after upd
ating the app. I use an xiaomi brand
94     try to work on this new upgrade.. our order for bike dont always get r
```

```
esponse
96     i notice that u always cancelled a long distance trip but the moment i
try to request for a short distance trip immediately a rider will be a availa
ble. if u know the promo is for short distance trip, u should specify so that
i will not waste my time requesting for ride.
97     The transport service is nice, the only issue is that they dont go to
far distance, once you request for it, the riders wont connect, until the loc
ation is changed. Apart from that its fast and reliable.
100    location accuracy and finder, willingness of bike man to go far locati
on and you really need much more advertisment.
101    I have been requesting for a ride since on Saturday last week 20/7/201
9, for more than 30 mins, keep seeing non is available, it's so annoying, rat
her I requested for others & got it one hand.
Name: Reviews, dtype: object
```

REVIEW INSIGHTS

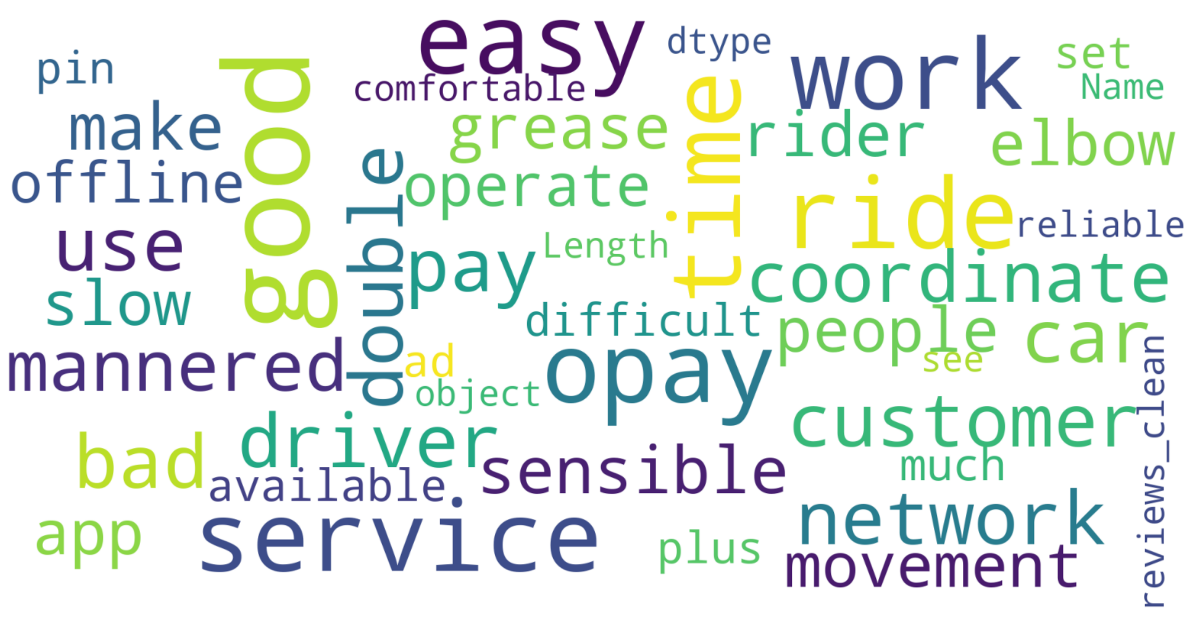
Now that we have our cleaned review data and have split the neutral ratings into good or bad categories. We can examine the text to see what insights we can gather.

Word Cloud

The word cloud is a visual representation of word frequency. We can immediately identify some key app services that seem to be important to customers, such as easy, good and opay. This method is somewhat controversial as it is difficult to interpret relative size (and therefore frequency) of words. It is also difficult to interpret context when isolated words are presented, such as in the case of "time, service, ride and work" which can be positive or negative.

In [86]:
```
wd_title = 'Opay App Reviews Word Cloud'
def show_wordcloud(data, title = wd_title):
    wordcloud = WordCloud(
        background_color = 'white',
        max_words = 200,
        max_font_size = 40,
        scale = 5,
        random_state = 52
    ).generate(str(data))

    fig = plt.figure(1, figsize = (20, 20))
    plt.axis('off')
    if title:
        fig.suptitle(title, fontsize = 30)
        fig.subplots_adjust(top = 1.4)

    plt.imshow(wordcloud)
    plt.show()
#show word cloud
show_wordcloud(df_class["reviews_clean"])
```

Opay App Reviews Word Cloud



N-Gram Analysis

N-grams are all the continuous sequence of words created from all the combinations of adjacent words in a text, with the variable n denoting the desired sequence length. By viewing sequences of text, we can overcome the shortcomings of the word clouds and draw some context from the common phrases seen in the review text.

The n_gram defined below creates a list of n-grams at the desired sequence length.

```
In [87]: def n_gram(token, n_gram, size ):
             tokenized = token.apply(lambda x: x.split())
             finder = BigramCollocationFinder.from_documents(tokenized.values)
             bigram_measures = nltk.collocations.BigramAssocMeasures()
             finder.apply_freq_filter(1)
             result = finder.nbest(bigram_measures.pmi, 10)
             ngram_list = [pair for row in tokenized for pair in ngrams(row, n_gram)]
             counts = Counter(ngram_list).most_common()
             print (pd.DataFrame.from_records(counts, columns=['gram', 'count']).head(s
         ize))
```

Taking an initial look at the n-grams for the entire cleaned corpus, as we can see noise still exist in our clean corpus our model classified the noise as a postive phrase i guess the model mistook is as "okay" phrase. Apart from the noise, the positive phrase "good, app, great, easy" this can also denote that the app is good and it is easy to use

lets take a look at the negative N-gram reviews, we can see mostly negative phrases which make somewhat sense considering that negative reviews tend to have more text. The most prevalent complaints being Server error issues, app unable to connect to the internet and inauccrate geolocation service,. These are possible areas for the app developers to address to improve customer satisfaction.

Positive N-Grams

Aside from the praise for the app, we can glimpse what customers like about opay app. the positive phrase "good, app, great, easy" this can also denote that the "app is good and it is easy to use". A successful app seems to be defined by the ability to make of it been easier, convenient and more accessible.

```
In [88]: df_best = df_class[(df_class['label'] == 1)]
         n_gram(df_best['reviews_clean'], 5, 15)
```

```
                                          gram  count
0   (ð⍰⍰⍰, ð⍰⍰⍰, ð⍰⍰⍰, ð⍰⍰⍰, ð⍰⍰⍰)           11
1   (one, best, app, ever, see)                4
2   (keep, tell, something, go, wrong)         4
3   (god, bless, opay, god, bless)             4
4   (best, app, i've, ever, use)               3
5   (ever, since, start, use, opay)            3
6   (great, app, easy, use, opay)              3
7   (always, tell, something, go, wrong)       2
8   (app, ever, keep, good, work)              2
9   (use, opay, make, life, easy)              2
10  (enroute, make, pay, service, render)      2
11  (nice, app, fast, payment, transaction)    2
12  (sometimes, take, long, get, rider)        2
13  (one, best, development, happen, nigeria)  2
14  (tell, set, high, accuracy, location)      2
```

Negative N-Grams

lets take a look at the negative N-gram reviews, we can see mostly negative phrases which make somewhat sense considering that negative reviews tend to have more text. The most prevalent complaints being Server error issues, app unable to connect to the internet and inauccrate geolocation service.

```
In [89]: df_worst = df_class[(df_class['label'] == 0)]
         n_gram(df_worst['reviews_clean'], 5, 20)
```

```
                                           gram  count
0   (keep, say, something, go, wrong)            23
1   (keep, tell, something, go, wrong)           19
2   (something, go, wrong, please, check)        18
3   (go, wrong, please, check, connection)       12
4   (say, something, go, wrong, check)           11
5   (something, go, wrong, check, internet)      11
6   (something, go, wrong, check, connection)    11
7   (tell, something, go, wrong, check)          10
8   (wrong, please, check, connection, try)      10
9   (go, wrong, check, internet, connection)     9
10  (always, say, something, go, wrong)          9
11  (keep, show, something, go, wrong)           8
12  (please, check, connection, try, later)      7
13  (app, say, something, go, wrong)             6
14  (error, message, something, go, wrong)       6
15  (set, high, accuracy, location, service)     5
16  (use, app, say, something, go)               5
17  (app, keep, tell, something, go)             5
18  (something, go, wrong, check, network)       5
19  (app, keep, say, something, go)              5
```

Predicted Positive N-gram

One interesting observation is the common occurrence of phrases like "good..", "easy..", etc. This seems to highlight some useful feedback for improving the apps and warrants a closer look.

In [90]:
```
n_gram(dfn_pred_good['reviews_clean'], 3, 15)
```

```
                                 gram  count
0   (make, life, easy)                5
1   (keep, good, work)                3
2   (pay, electricity, bill)          3
3   (sometimes, take, time)           3
4   (still, need, improvement)        3
5   (rider, accept, ride)             3
6   (make, transportation, easy)      3
7   (nice, easy, use)                 3
8   (app, great, service)             2
9   (great, experience, far)          2
10  (use, app, thank)                 2
11  (app, good, easy)                 2
12  (good, easy, use)                 2
13  (good, one, fast)                 2
14  (take, like, forever)             2
```

In [91]:
```
dfn_pg= dfn_pred_good[dfn_pred_good['reviews_clean'].str.contains("easy")]
#dfn_pg['Reviews'].head(10)
dfn_pg['reviews_clean'].head(10)
```

Out[91]:
```
65      okay previous complaint resolve chaging rating sincerely like app try
achieve platform expense easy convenient personalize affordable might mvp min
imum viable product advise app work customer centre response time gui interfa
ce app merchant rider also monitor hopefully really work
69      easy book ride
153     easy reliable
207     easy-going
271     app content interest easy use look forward ofood deal
278     hello app good easy use think app say ride price ridiculously increase
hundred naira continiues like do use app thank
302     ok easy use
304     good easy
312     cool easy
375     easy ride
Name: reviews_clean, dtype: object
```

Predicted Negative N-grams

many of users complained of experiencing technical issues of the app not able to connect to the internet on their mobile phones, also geo-location for oride service not accurate

```
In [92]: n_gram(dfn_pred_bad['reviews_clean'], 4, 15)
```

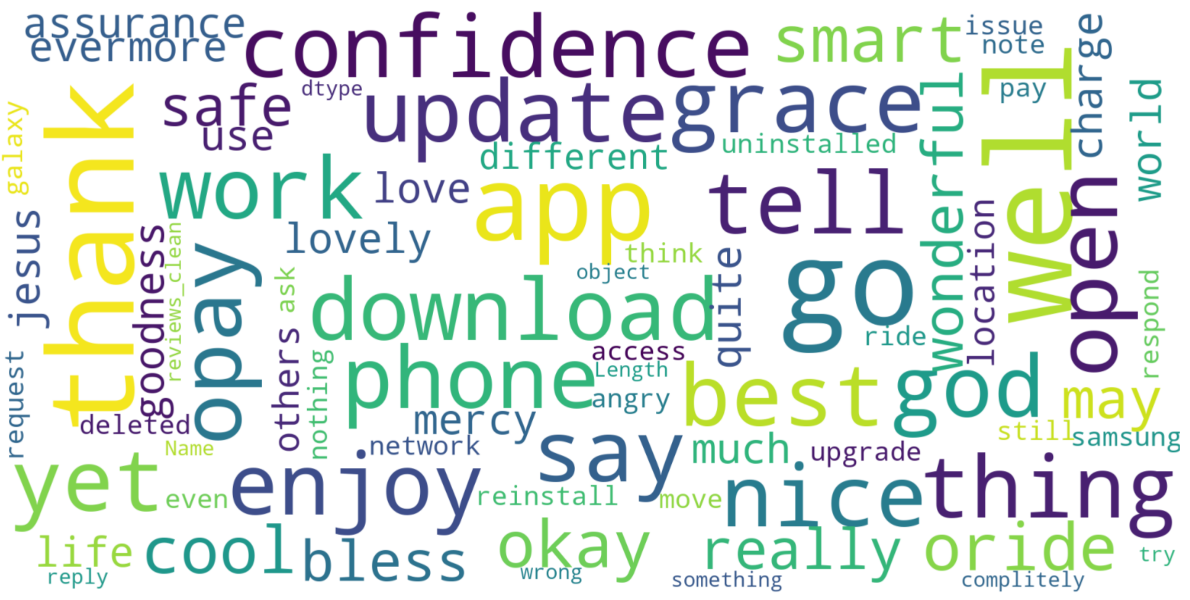
```
                                   gram  count
0   (keep, say, something, go)         4
1   (say, something, go, wrong)        4
2   (set, high, accuracy, location)    4
3   (show, something, go, wrong)       3
4   (please, set, high, accuracy)      3
5   (say, internal, server, error)     3
6   (call, customer, care, line)       3
7   (transfer, money, bank, account)   2
8   (something, go, wrong, check)      2
9   (take, long, time, get)            2
10  (check, connection, try, later)    2
11  (refuse, open, keep, say)          2
12  (open, keep, say, something)       2
13  (something, go, wrong, please)     2
14  (go, wrong, please, check)         2
```

More Word Clouds

In [93]:
```
df_highest = df_class[df_class["num_words"] >=8].sort_values("pos", ascending
= False)[["reviews_clean", "pos"]]
print("Positive Review Preview")
print(df_highest['reviews_clean'].head(10))
# print wordcloud
wd_title = 'Positive Reviews Word Cloud'
show_wordcloud(df_highest["reviews_clean"], title = wd_title)
```

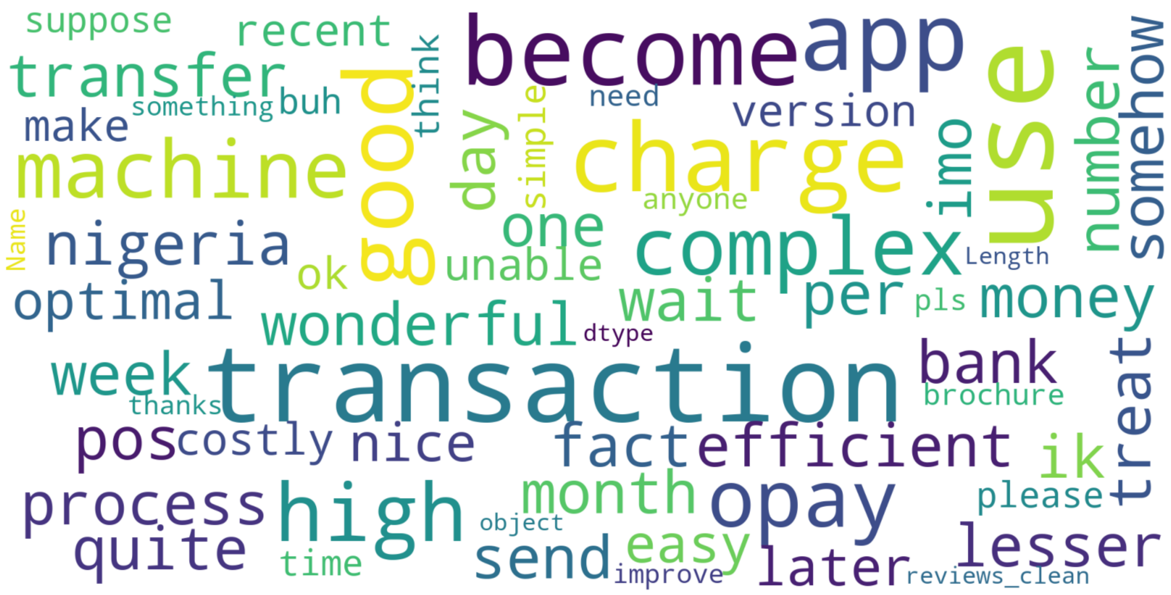
```
Positive Review Preview
12460    well okay oride go well confidence smart cool grace
10832    thanks app really enjoy nice wonderful best ever
13779    well well opay nice safe thank opay may god bless
28278    assurance confidence world best app use lovely love download enjoy
35328    thank god goodness life grace mercy evermore thank jesus
1468     love interest life save project god bless initiator(s
39385    smart easy use god bless opay love opay
12170    awesome actually nice good opportunity really save money super magni
ficent app â□¤ï¸□ð□□□
16603    beautiful experience thank god something like make transportation ea
sy god bless
31985    really appreciate much enjoy god bless much thank
Name: reviews_clean, dtype: object
```

Positive Reviews Word Cloud



```
In [94]: # print wordcloud
         wd_title = 'Predicted Positive Reviews Word Cloud'
         show_wordcloud(dfn_pred_good['reviews_clean'], title = wd_title)
```

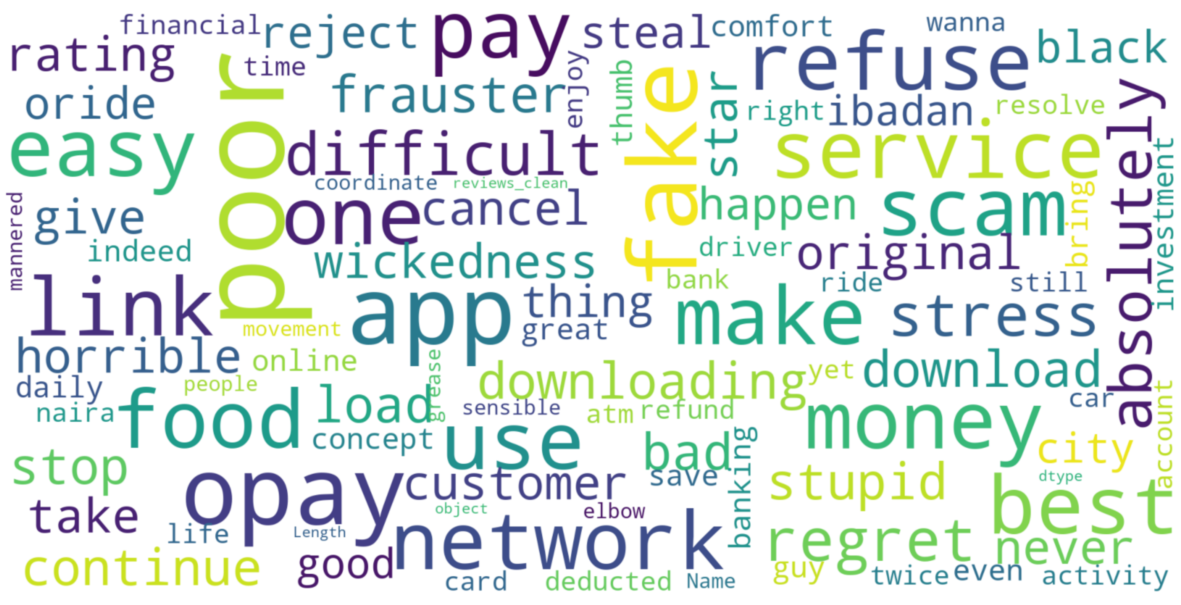
Predicted Positive Reviews Word Cloud



In [95]:
```python
# lowest negative sentiment reviews (with more than 5 words)
df_lowest = df_class[df_class["num_words"] >= 8].sort_values("neg", ascending
= False)[["reviews_clean", "neg"]]
print('Negative Review Preview')
print(df_lowest['reviews_clean'].head(10))

# print wordcloud
wd_title = 'Negative Reviews Word Cloud'
show_wordcloud(df_lowest["reviews_clean"], title = wd_title)
```

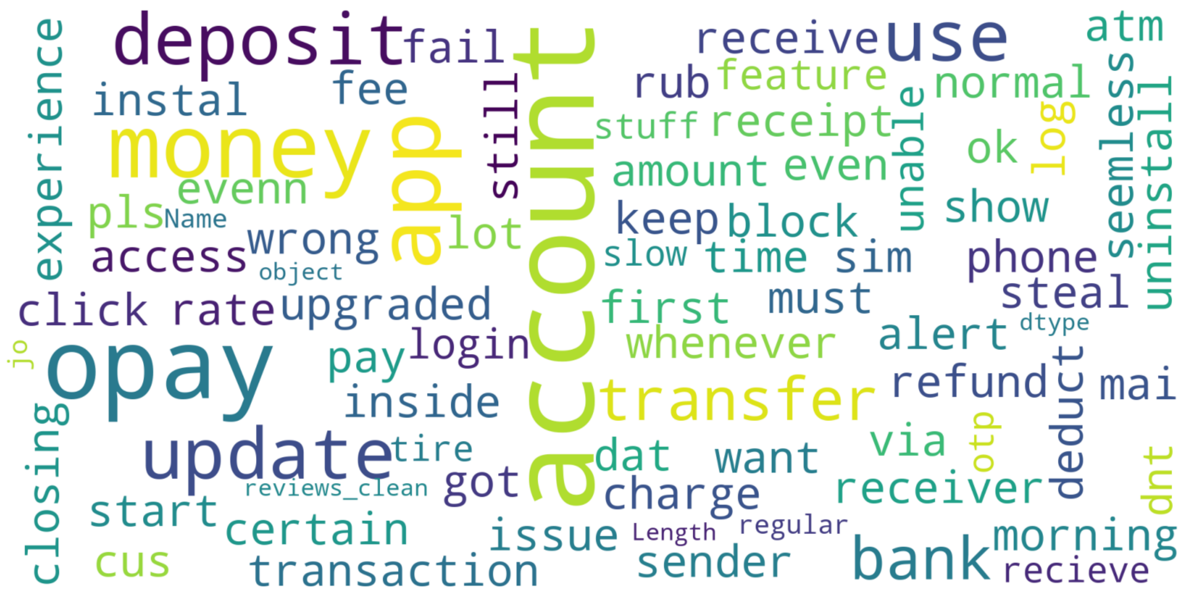
```
Negative Review Preview
7071     difficult pay stress pay app absolutely regret downloading
4600     app refuse load,while download stop refuse continue bad
112      stupid network fake network frauster use fake fake original wickedne
ss
42777    horrible star rating i'd give app poor customer service poor service
poor poor poor
10836    scam never food take get food cancel reject steal money scam black
24125    bad baddo baddest regret work orider ilorin ahahhaha slavery mean mo
dern slavery
1448     fake app network connection waste mb useless app
20357    terrible log let alone register battle week keep send verification c
ode wrong incompetent
42772    rider bad didnt pick order ask pay bad
12331    people really frustrated rider keep reject order really disappointed
Name: reviews_clean, dtype: object
```

Negative Reviews Word Cloud



In [96]:
```
# print wordcloud
wd_title = 'Predicted Negative Reviews Word Cloud'
show_wordcloud(dfn_pred_bad['reviews_clean'], title = wd_title)
```

Predicted Negative Reviews Word Cloud



In [ ]: