09_Amazon_Food_Reviews_Ensemble

April 14, 2019

1 Amazon Fine Food Reviews Analysis

Data Source: https://www.kaggle.com/snap/amazon-fine-food-reviews

EDA: https://nycdatascience.com/blog/student-works/amazon-fine-foods-visualization/

The Amazon Fine Food Reviews dataset consists of reviews of fine foods from Amazon.

Number of reviews: 568,454 Number of users: 256,059 Number of products: 74,258 Timespan: Oct 1999 - Oct 2012 Number of Attributes/Columns in data: 10

Attribute Information:

1. Id
2. ProductId - unique identifier for the product
3. UserId - unqiue identifier for the user
4. ProfileName
5. HelpfulnessNumerator - number of users who found the review helpful
6. HelpfulnessDenominator - number of users who indicated whether they found the review helpful or not
7. Score - rating between 1 and 5
8. Time - timestamp for the review
9. Summary - brief summary of the review
10. Text - text of the review

Objective: Given a review, determine whether the review is positive (rating of 4 or 5) or negative (rating of 1 or 2).

[Q] How to determine if a review is positive or negative? [Ans] We could use Score/Rating. A rating of 4 or 5 can be cosnidered as a positive review. A rating of 1 or 2 can be considered as negative one. A review of rating 3 is considered nuetral and such reviews are ignored from our analysis. This is an approximate and proxy way of determining the polarity (positivity/negativity) of a review.

2 Random Forest

```
<li><strong>Apply Random Forests & GBDT on these feature sets</strong>
    <ul>
        <li><font color='red'>SET 1:</font>Review text, preprocessed one converted into vectors
        <li><font color='red'>SET 2:</font>Review text, preprocessed one converted into vectors
        <li><font color='red'>SET 3:</font>Review text, preprocessed one converted into vectors
        <li><font color='red'>SET 4:</font>Review text, preprocessed one converted into vectors
```

```
    </ul>
</li>
<br>
<li><strong>The hyper paramter tuning (Consider two hyperparameters: n_estimators & max_depth)</
    <ul>
<li>Find the best hyper parameter which will give the maximum <a href='https://www.appliedaicour
<li>Find the best hyper paramter using k-fold cross validation or simple cross validation data</
<li>Use gridsearch cv or randomsearch cv or you can also write your own for loops to do this tas
    </ul>
</li>
<br>
<li><strong>Feature importance</strong>
    <ul>
<li>Get top 20 important features and represent them in a word cloud. Do this for BOW & TFIDF.</
    </ul>
</li>
<br>
<li><strong>Feature engineering</strong>
    <ul>
<li>To increase the performance of your model, you can also experiment with with feature enginee
        <ul>
        <li>Taking length of reviews as another feature.</li>
        <li>Considering some features from review summary as well.</li>
    </ul>
    </ul>
</li>
<br>
<li><strong>Representation of results</strong>
    <ul>
<li>You need to plot the performance of model both on train data and cross validation data for e
<img src='3d_plot.JPG' width=500px> with X-axis as <strong>n_estimators</strong>, Y-axis as <str
        <p style="text-align:center;font-size:30px;color:red;"><strong>(or)</strong></p> <br>
<li>You need to plot the performance of model both on train data and cross validation data for e
<img src='heat_map.JPG' width=300px> <a href='https://seaborn.pydata.org/generated/seaborn.heatm
<li>You choose either of the plotting techniques out of 3d plot or heat map</li>
<li>Once after you found the best hyper parameter, you need to train your model with it, and fin
<img src='train_test_auc.JPG' width=300px></li>
<li>Along with plotting ROC curve, you need to print the <a href='https://www.appliedaicourse.co
<img src='confusion_matrix.png' width=300px></li>
    </ul>
</li>
<br>
<li><strong>Conclusion</strong>
    <ul>
<li>You need to summarize the results at the end of the notebook, summarize it in the table form
    <img src='summary.JPG' width=400px>
</li>
    </ul>
```

#
Note: Data Leakage

1. There will be an issue of data-leakage if you vectorize the entire data and then split it into train/cv/test.
2. To avoid the issue of data-leakag, make sure to split your data first and then vectorize it.
3. While vectorizing your data, apply the method fit_transform() on you train data, and apply the method transform() on cv/test data.
4. For more details please go through this link.

2.1 [A] Applying RF

3 Import Required Packages

In [1]:
```
import os
from datetime import datetime
import pandas as pd
import numpy as np

# visualization related packages
import matplotlib.pyplot as plt
import seaborn as sns
sns.set()

# data prerocessing related
from sklearn.preprocessing import StandardScaler

# import model related packages and visualization of trees
from sklearn.ensemble import RandomForestClassifier
from xgboost import XGBClassifier
from sklearn.tree import export_graphviz
import graphviz


# import model selection packages
from sklearn.model_selection import StratifiedKFold

# import model evaluation related packages
from sklearn.metrics import confusion_matrix
from sklearn.metrics import precision_recall_fscore_support
from sklearn.metrics import auc, roc_curve
from scipy import interp

# calibration related package
from sklearn.calibration import CalibratedClassifierCV

# visualization related packages
from wordcloud import WordCloud
```

```
from prettytable import PrettyTable

# for forming grid
from itertools import product
```

4 UTILS functions

4.1 Data Preprocessing related Functions

```
In [2]: def preprocess_data(config_dict, scaling=True, dim_reduction=False):
            """
            This function does preprocessing of data such as column standardization and
            dimensionality reduction using Truncated SVD
            """

            # Read train, test data frames & truncate it as needed
            train_df = pd.read_csv(config_dict['train_csv_path'], index_col=False)
            train_df = train_df.iloc[0:config_dict['train_size']]
            test_df = pd.read_csv(config_dict['test_csv_path'], index_col=False)
            test_df = test_df.iloc[0:config_dict['test_size']]

            # print the statisics of train, test df
            print('Train df shape',train_df.shape)
            print('Class label distribution in train df:\n', train_df['Label'].value_counts())
            print('Test df shape',test_df.shape)
            print('Class label distribution in test df:\n', test_df['Label'].value_counts())

            # separate features and labels
            train_features = train_df.drop(['Label', 'Id'], axis=1)
            train_labels = train_df['Label']
            test_features = test_df.drop(['Label', 'Id'], axis=1)
            test_labels = test_df['Label']

            # set feature names
            feature_name_list = train_features.columns.values.tolist()

            # If Scaling is opted scale the train, test data
            if scaling:
                standard_scaler = StandardScaler()
                standard_scaler.fit(train_features)

                train_features = pd.DataFrame(standard_scaler.transform(train_features),
                                              columns=feature_name_list)
                test_features = pd.DataFrame(standard_scaler.transform(test_features),
                                             columns=feature_name_list)

            print('Shape of ->  train features :%d,%d, test features: %d,%d'%(train_features.sha
```

```
                                                                    test_features.shap
print('Shape of -> train labels :%d, test labels: %d'%(train_labels.shape[0],
                                                        test_labels.shape[0],))

# if dim reduction is opted, reduce the dimension
if dim_reduction:
    # create an SVD object
    truc_svd = TruncatedSVD(n_components=train_features.shape[1]-1, n_iter=8, algori

    # fit to data
    truc_svd.fit(train_features)

    # get explained variance ratio of each component
    explained_var_ratios = truc_svd.explained_variance_ratio_

    # get cummulative ratio list for selecting the number of components
    cumulative_ratios = np.cumsum(explained_var_ratios)

    # plot the #components vs captured variance in the data
    plt.title('SVD Decomposition')
    plt.xlabel('Number of components')
    plt.ylabel('Cumulative Percentage Ratio')
    plt.plot(range(1, len(cumulative_ratios) + 1), cumulative_ratios)
    plt.show()

    # set a threshold for stopping selection of components.
    svd_thesh = 0.001
    # select the number of components as the first component for which the differenc
    #  very less (less than svd thresh) compared with the very next component
    selected_dim = list(filter(lambda x : x[1] < svd_thesh, enumerate(np.diff(cumula
    print('Num dimensions selected by SVD', selected_dim)
    print('Total variance captured:%f'%(cumulative_ratios[selected_dim]))

    # create an object for selecting the components
    truc_svd = TruncatedSVD(n_components=selected_dim, n_iter=8, algorithm='randomiz
    # refit with the desired number of components
    truc_svd.fit(train_features)

    # reduce the number of dimensions to selected number of components
    train_features = pd.DataFrame(truc_svd.transform(train_features))
    test_features = pd.DataFrame(truc_svd.transform(test_features))

    # get the shape of final data frame and print it
    size_tuple = train_features.shape + test_features.shape
    print('Shape of train df:(%d,%d), Test DF:(%d,%d)'%size_tuple)


return (train_features, train_labels, test_features, test_labels,)
```

4.2 Model Training and Evaluation related packages

```
In [3]: def plot_roc_curves_single(fold_prediction_list, plot_title):
            """
            This function helps to plot the ROC curve for a set of predictions on different fold
            """

            # set figure size
            plt.figure(figsize=(10,10))

            # reference points for X axis
            ref_points = np.linspace(0.0, 1.0, 100)

            # two lists for auc values and tpr rates
            auc_scores_list = list()
            tpr_list = list()

            # plot ROC curve for each fold
            for index, (actual_probs, predicted_probs,) in enumerate(fold_prediction_list):

                # compute ROC curve and get the AUC value for this fold
                fpr, tpr, thresholds = roc_curve(actual_probs, predicted_probs)
                # compute AUC
                auc_score = auc(fpr, tpr)

                # interpolation to approximate the curve
                tp_rates = interp(ref_points, fpr, tpr)
                tp_rates[0] = 0.0 # for setting the bottom left point

                # for plotting the individual fold and finding the average
                auc_scores_list.append(auc_score)
                tpr_list.append(tp_rates)

                # plot this fold info into a fig
                plt.plot(fpr, tpr, alpha=0.3, lw=1, label='Test AUC for fold %d : %f'%(index, au

            # Plot the random classifier
            plt.plot([0,1],[0,1], alpha=0.8, linestyle='--', color='red', label='Random Guess',

            # Plot the mean performance
            mean_tpr = np.mean(tpr_list, axis=0)
            std_tprs = np.std(tpr_list, axis=0)
            # mean value of AUC and its standard deviation
            mean_auc = auc(ref_points, mean_tpr)
            std_auc = np.std(auc_scores_list)

            plt.plot(ref_points, mean_tpr, linestyle='-', color='g', lw=2,
                     alpha=0.8, label='Test Mean AUC %f $\pm$ %f'%(mean_auc,std_auc))
```

```
        # Find upper and lower bounds for shading the region around TPRs
        tprs_lower_region = np.maximum(mean_tpr - std_tprs, 0)
        tprs_upper_region = np.minimum(mean_tpr + std_tprs, 1)

        # Fill the region between upper and lower in gray color
        plt.fill_between(ref_points, tprs_lower_region, tprs_upper_region, color='gray', alp
                        label='Test Around the mean TPRs')

        # arange the plot
        plt.xlim([-0.05, 1.05])
        plt.ylim([-0.05, 1.05])
        plt.xlabel('False Positive Rates')
        plt.ylabel('True Positive Rates')
        plt.title('ROC Curve - ' + plot_title)
        plt.legend(loc='lower right')
        plt.show()

        return mean_auc
```

```
In [4]: def plot_roc_curves_pair(train_fold_prediction_list, validation_fold_prediction_list, pl
        """
        This function helps to plot the ROC curve for a set of predictions on different fold
        """

        # set figure size
        plt.figure(figsize=(10,10))

        # reference points for X axis
        ref_points = np.linspace(0.0, 1.0, 100)


        # -------------------- 1 FOR TRAIN -------------------------------------------
        # two lists for auc values and tpr rates
        auc_scores_list = list()
        tpr_list = list()

        # plot ROC curve for each fold
        for index, (actual_probs, predicted_probs,) in enumerate(train_fold_prediction_list)

            # compute ROC curve and get the AUC value for this fold
            fpr, tpr, thresholds = roc_curve(actual_probs, predicted_probs)
            # compute AUC
            auc_score = auc(fpr, tpr)

            # interpolation to approximate the curve
            tp_rates = interp(ref_points, fpr, tpr)
```

```
    tp_rates[0] = 0.0 # for setting the bottom left point

    # for plotting the individual fold and finding the average
    auc_scores_list.append(auc_score)
    tpr_list.append(tp_rates)

    # plot this fold info into a fig
    plt.plot(fpr, tpr, alpha=0.3, lw=1, label='Train AUC for fold %d : %f'%(index, a

# Plot the mean performance
mean_tpr = np.mean(tpr_list, axis=0)
std_tprs = np.std(tpr_list, axis=0)
# mean value of AUC and its standard deviation
mean_auc = auc(ref_points, mean_tpr)
std_auc = np.std(auc_scores_list)

plt.plot(ref_points, mean_tpr, linestyle='-', color='b', lw=2,
         alpha=0.8, label='Train Mean AUC %f $\pm$ %f'%(mean_auc,std_auc))


# Find upper and lower bounds for shading the region around TPRs
tprs_lower_region = np.maximum(mean_tpr - std_tprs, 0)
tprs_upper_region = np.minimum(mean_tpr + std_tprs, 1)

# Fill the region between upper and lower in gray color
plt.fill_between(ref_points, tprs_lower_region, tprs_upper_region, color='b', alpha=
                label='Train Around the mean TPRs')

# train mean auc
train_mean_auc = mean_auc


# -------------------- 1 FOR VALIDATION ----------------------------------------
# two lists for auc values and tpr rates
auc_scores_list = list()
tpr_list = list()

# plot ROC curve for each fold
for index, (actual_probs, predicted_probs,) in enumerate(validation_fold_prediction_

    # compute ROC curve and get the AUC value for this fold
    fpr, tpr, thresholds = roc_curve(actual_probs, predicted_probs)
    # compute AUC
    auc_score = auc(fpr, tpr)

    # interpolation to approximate the curve
    tp_rates = interp(ref_points, fpr, tpr)
    tp_rates[0] = 0.0 # for setting the bottom left point
```

```
        # for plotting the individual fold and finding the average
        auc_scores_list.append(auc_score)
        tpr_list.append(tp_rates)

        # plot this fold info into a fig
        plt.plot(fpr, tpr, alpha=0.3, lw=1, label='Val AUC for fold %d : %f'%(index, auc

    # Plot the mean performance
    mean_tpr = np.mean(tpr_list, axis=0)
    std_tprs = np.std(tpr_list, axis=0)
    # mean value of AUC and its standard deviation
    mean_auc = auc(ref_points, mean_tpr)
    std_auc = np.std(auc_scores_list)


    # val mean auc
    val_mean_auc = mean_auc

    plt.plot(ref_points, mean_tpr, linestyle='-', color='g', lw=2,
             alpha=0.8, label='Val Mean AUC %f $\pm$ %f'%(mean_auc,std_auc))


    # Find upper and lower bounds for shading the region around TPRs
    tprs_lower_region = np.maximum(mean_tpr - std_tprs, 0)
    tprs_upper_region = np.minimum(mean_tpr + std_tprs, 1)

    # Fill the region between upper and lower in gray color
    plt.fill_between(ref_points, tprs_lower_region, tprs_upper_region, color='g', alpha=
                     label='Val Around the mean TPRs')

    # Plot the random classifier
    plt.plot([0,1],[0,1], alpha=0.8, linestyle='--', color='red', label='Random Guess',

    # arange the plot
    plt.xlim([-0.05, 1.05])
    plt.ylim([-0.05, 1.05])
    plt.xlabel('False Positive Rates')
    plt.ylabel('True Positive Rates')
    plt.title('ROC - ' + plot_title)
    plt.legend(loc='lower right')
    plt.show()

    return (train_mean_auc, val_mean_auc)
```

```
In [5]: def get_confusion_matrix(actual_list, predicted_list, cm_title):
            """
            This function plots the confusion matrix given ground truth and predicted
```

```
    """

    conf_matrix = confusion_matrix(actual_list, predicted_list)
    col_names = ['Negative', 'Positive']
    conf_df = pd.DataFrame(conf_matrix,columns=col_names)
    conf_df.index = col_names

    plt.figure(figsize = (5,5))

    plt.title(cm_title)
    sns.set(font_scale=1.4)#for label size
    sns.heatmap(conf_df, annot=True, annot_kws={"size": 16}, fmt='g')

    plt.show()
```

```
In [6]: def create_hyperparam_heatmap(hyper_param_score_list, hamp_title):
            """
            This function accepts a list containing the hyper parameters ('max_depth', 'n_estima
            the corresponding score value, such as AUC, FScore etc and plot the heatmap.
            """

            hyp_score_list = [item[1] for item in hyper_param_score_list]
            coord_list = [item[0] for item in hyper_param_score_list]
            hyp_df = pd.DataFrame(coord_list, columns=['max_depth', 'n_estimators'])
            hyp_df['AUC'] = hyp_score_list

            # pivot the table for heatmap representation
            pivoted_hyp = pd.pivot_table(hyp_df, index='max_depth', columns='n_estimators',
                                         values='AUC', fill_value=0)
            sns.heatmap(data=pivoted_hyp, annot=True)
            plt.title(hamp_title)
            plt.show()
```

```
In [7]: def find_best_hyperparameter(config_dict, train_features, train_labels):
            """
            This function helps to find the best hyper parameters (alpha) for RF or XGB algorith
            All set of hyper param values using which the model to be evaluated can be passed to
            list hyperparam_list.
            """
            print('='*100)

            stratified_partition = StratifiedKFold(n_splits=2)

            # read some config settings
            hyperparam_list = config_dict['hyperparam_list']
            implementation = config_dict['implementation']

            # declare a hyper param list
```

```
hyper_param_score_list = list()

for hyp_val in hyperparam_list:

    # declare three lists for holding prediction informations
    # for train set performance
    train_actual_labels_list = list()
    train_predicted_probs_list = list()
    train_predicted_labels_list = list()

    # for validation set performance
    val_actual_labels_list = list()
    val_predicted_probs_list = list()
    val_predicted_labels_list = list()

    # Model defined here
    if implementation == 'rf':
        classifier = RandomForestClassifier(max_depth=hyp_val[0], n_estimators=hyp_v
        # Declare a calibrated classifer
        calib_classifier = CalibratedClassifierCV(base_estimator=classifier,
                                                  method='isotonic',
                                                  cv='prefit')
    else:
        classifier = XGBClassifier(max_depth=hyp_val[0], n_estimators=hyp_val[1])



    # Train the model and evaluate it on the current fold data
    for train_indices, val_indices in stratified_partition.split(train_features, tra

        # A) train the model suing StratifiedKFold method

        # get the train features, train labels for this fold
        train_feat_data = train_features.iloc[train_indices, :]
        train_label_data = train_labels[train_indices]

        # train the classifier
        classifier.fit(train_feat_data, train_label_data)

        # For random forest we need to calibrate
        if implementation == 'rf':
            calib_classifier.fit(train_feat_data, train_label_data)
            train_eval_y_probs = calib_classifier.predict_proba(train_feat_data)[:,
        else:
            train_eval_y_probs = classifier.predict_proba(train_feat_data)[:, 1]

        # estimate the training metrics on (train fold)
        train_eval_y_value = classifier.predict(train_feat_data)
```

```
        # save the results for ROC plot
        train_actual_labels_list.append(train_label_data)
        train_predicted_probs_list.append(train_eval_y_probs)
        train_predicted_labels_list.append(train_eval_y_value)


        # B) predict the labels and probability for this fold (validation fold)

        # get the validation features, validation labels for this fold
        validation_feat_data = train_features.iloc[val_indices, :]
        validation_label_data = train_labels[val_indices]

        # evaluate the classifier on validation set
        val_actual_labels_list.append(validation_label_data)

        # predict the probability for validation data
        if implementation == 'rf':
            # validation step
            val_eval_y_probs = calib_classifier.predict_proba(validation_feat_data)[
        else:
            val_eval_y_probs = classifier.predict_proba(validation_feat_data)[:, 1]

        # predict the output for validation
        val_eval_y_value = classifier.predict(validation_feat_data)

        # save the results for ROC plot
        val_predicted_probs_list.append(val_eval_y_probs)
        val_predicted_labels_list.append(val_eval_y_value)


    # plot the results to select best hyper params

    # train data plot
    train_fold_prediction_list = list(zip(train_actual_labels_list, train_predicted_

    # plot the roc curve for train data
    if len(hyp_val) == 2:

        # get input data for plotting train and validation
        train_fold_prediction_list = list(zip(train_actual_labels_list, train_predic
        val_fold_prediction_list = list(zip(val_actual_labels_list, val_predicted_pr

        # validation data plot
        mean_aucs = plot_roc_curves_pair(train_fold_prediction_list, val_fold_predic
            'ROC Plot: Train & Validation of Ensemble (max_depth:%d, num_estimators:

        # update the list with the scores for this hyperparam for both tain, validat
```

```
            hyper_param_score_list.append((hyp_val, mean_aucs[0], mean_aucs[1]))
        else:
            print('Invalid hyper param configs')

    print('='*100)

    return hyper_param_score_list
```

In [8]:
```
def train_and_validate_model(config_dict, train_features, train_labels):
    """
    This function train a model, validate it using cross validation and return the best
    obtained during cross validation.
    """

    # get model is RF or XGB
    implementation = config_dict['implementation']

    # find best hyperparameter by stratified cross validation
    hyper_param_scores_list = find_best_hyperparameter(config_dict, train_features, trai

    # extract individual fields
    depth_value_list = [item[0][0] for item in hyper_param_scores_list]
    num_estimators_list = [item[0][1] for item in hyper_param_scores_list]
    tr_auc_list = [item[1] for item in hyper_param_scores_list]
    val_auc_list = [item[2] for item in hyper_param_scores_list]

    # print k vs auc
    print('+'*100)
    print('\n\n The depth vs AUC score plot')
    plt.plot(depth_value_list, tr_auc_list, label='Train AUC')
    plt.plot(depth_value_list, val_auc_list, label='Validation AUC')
    plt.xlabel('depth')
    plt.ylabel('AUC Scores')
    plt.title('depth vs AUC')
    plt.legend()
    plt.show()
    print('+'*100)

    print('+'*100)
    print('\n\n The num_estimators vs AUC score plot')
    plt.plot(num_estimators_list, tr_auc_list, label='Train AUC')
    plt.plot(num_estimators_list, val_auc_list, label='Validation AUC')
    plt.xlabel('num_estimatores')
    plt.ylabel('AUC Scores')
    plt.title('num_estimators vs AUC')
    plt.legend()
    plt.show()
    print('+'*100)
```

```
# heatmap plot for selecting the best hyper params
hyper_param_scores_list_train = [(item[0], item[1]) for item in hyper_param_scores_l
hyper_param_scores_list_validation = [(item[0], item[2]) for item in hyper_param_sco

# plot heatmap for train and validation
create_hyperparam_heatmap(hyper_param_scores_list_train, 'Hyper param Heatmap DT-Tra
create_hyperparam_heatmap(hyper_param_scores_list_validation, 'Hyper param Heatmap D


#Set the best Hyper param based on above plots
best_hyper_param = min(hyper_param_scores_list, key=lambda x: abs(x[1] - x[2]) + (1
print('Best hyperparam value: ', best_hyper_param)

# decalre the optional calib classifier as None
calib_classifier = None

# Final Model defined here
if implementation == 'rf':
    classifier = RandomForestClassifier(max_depth=best_hyper_param[0],
                                        n_estimators=best_hyper_param[1])
    # Declare a calibrated classifer
    calib_classifier = CalibratedClassifierCV(base_estimator=classifier,
                                        method='isotonic',
                                        cv='prefit')
else:
    classifier = XGBClassifier(max_depth=best_hyper_param[0],
                               n_estimators=best_hyper_param[1])

# train the classifier
classifier.fit(train_features, train_labels)

if implementation == 'rf':
    calib_classifier.fit(train_features, train_labels)
    train_eval_y_probs = calib_classifier.predict_proba(train_features)[:, 1]
else:
    train_eval_y_probs = classifier.predict_proba(train_features)[:, 1]

# estimate the training metrics on (train fold)
train_eval_y_value = classifier.predict(train_features)
train_info_list = list(zip([train_labels], [train_eval_y_probs]))


# get train auc value
train_auc_val = plot_roc_curves_single(train_info_list,
                                 'Final Model DT (depth:%d, n_estimators:%d)'%best_hy
```

```
    # print the confusion matrix
    get_confusion_matrix(train_labels, train_eval_y_value, 'Train Confusion Matrix')

    # compute precision and other matric
    all_metrics = precision_recall_fscore_support(train_labels, train_eval_y_value)
    all_metrics_df = pd.DataFrame(list(all_metrics), columns=['Negative', 'Positive'])
    all_metrics_df.index = ['Precision', 'Recall', 'Fscore', 'Support']
    # convert fscore to percentage
    fscores = all_metrics[2] * 100.0

    print(all_metrics_df)

    # return the final model
    return (classifier, calib_classifier,)
```

```
In [9]: def test_and_evaluate_model(config_dict, model, test_features, test_labels):
    """
    This function test and evaluate the performance on unseen data.
    """

    # separate out models
    classifier = model[0]
    calib_classifier = model[1]

    # get type of model
    implementation = config_dict['implementation']

    # evaluate final model on test dataset
    if implementation == 'xgb':
        test_eval_y_probs = classifier.predict_proba(test_features)[:, 1]
    else:
        test_eval_y_probs = calib_classifier.predict_proba(test_features)[:, 1]

    test_eval_y_value = classifier.predict(test_features)
    test_info_list = list(zip([test_labels], [test_eval_y_probs]))

    # test auc value
    best_hyp = (classifier.max_depth, classifier.n_estimators,)
    test_auc_val = plot_roc_curves_single(test_info_list,
                                          'Final Model DT (depth:%d, n_estimators:%d)'%best_hyp

    print('Test auc score ', test_auc_val)

    # print the confusion matrix
    get_confusion_matrix(test_labels, test_eval_y_value, 'Test Confusion Matrix')

    # compute precision and other matric
```

```
    all_metrics = precision_recall_fscore_support(test_labels, test_eval_y_value)
    all_metrics_df = pd.DataFrame(list(all_metrics), columns=['Negative', 'Positive'])
    all_metrics_df.index = ['Precision', 'Recall', 'Fscore', 'Support']
    # convert fscore to percentage
    fscores = all_metrics[2] * 100.0

    # print all the metrics
    print(all_metrics_df)

    # create an entry for pretty table using all the metrics

    # get the table entry
    ptabe_entry = [best_hyp] + [ '%.4f'%item for item in ([test_auc_val] + list(fscores)

    print('Results Summary: \n', list(zip(['Hyper Param', 'AUC', 'f-score(-ve)', 'f-scor
                                          ptabe_entry)))

    return ptabe_entry
```

4.2.1 [A.1] Applying Random Forests on BOW, SET 1

In [10]:
```
# form two lists
depth_list = [3, 5, 10, 20, 50] # depends on size of dataset
n_estimators_list = [20, 60, 120, 300, 500] # depends on size of dataset

# declare a configuartion dictionary
config_dict = {
    'train_csv_path' : '/home/amd_3/AAIC/ASM_REPO/Processed_data/AMZN_FOOD_REVIW/BOW/tr
    'test_csv_path' : '/home/amd_3/AAIC/ASM_REPO/Processed_data/AMZN_FOOD_REVIW/BOW/tes
    'train_size' : 50000,
    'test_size' : 20000,
    'hyperparam_list' : list(product(depth_list, n_estimators_list)),
    'implementation': 'rf' # 'xgb' or 'rf'
}
```

In [11]:
```
# read the train, test data and preprocess it
train_features, train_labels, test_features, test_labels = preprocess_data(config_dict,
                                                                           scaling=True
                                                                           dim_reductio

# create a list containing all the features
feature_name_list = train_features.columns.values.tolist()

# train and validate the model
model = train_and_validate_model(config_dict, train_features, train_labels)

# test and evaluate the model
ptabe_entry_a1 = test_and_evaluate_model(config_dict, model, test_features, test_labels
```

```
Train df shape (50000, 503)
Class label distribution in train df:
 0    25029
1    24971
Name: Label, dtype: int64
Test df shape (20000, 503)
Class label distribution in test df:
 1    16520
0     3480
Name: Label, dtype: int64
Shape of ->  train features :50000,501, test features: 20000,501
Shape of -> train labels :50000, test labels: 20000
==========================================================================================
```

ROC - ROC Plot: Train & Validation of Ensemble (max_depth:3, num_estimators:60)
True Positive Rates
False Positive Rates
0.0
0.2
0.4
0.6
0.8
1.0
Train AUC for fold 0 : 0.853001
Train AUC for fold 1 : 0.843000
Train Mean AUC 0.847643 ± 0.005001
Val AUC for fold 0 : 0.852717
Val AUC for fold 1 : 0.835390
Val Mean AUC 0.843750 ± 0.008663
Random Guess
Train Around the mean TPRs
Val Around the mean TPRs

ROC - ROC Plot: Train & Validation of Ensemble (max_depth:3, num_estimators:120)
True Positive Rates
False Positive Rates
0.0
0.2
0.4
0.6
0.8
1.0
Train AUC for fold 0 : 0.860226
Train AUC for fold 1 : 0.862374
Train Mean AUC 0.860816 ± 0.001074
Val AUC for fold 0 : 0.860414
Val AUC for fold 1 : 0.853759
Val Mean AUC 0.856650 ± 0.003327
Random Guess
Train Around the mean TPRs
Val Around the mean TPRs

ROC - ROC Plot: Train & Validation of Ensemble (max_depth:3, num_estimators:300)
True Positive Rates
False Positive Rates
0.0
0.2
0.4
0.6
0.8
1.0
Train AUC for fold 0 : 0.860956
Train AUC for fold 1 : 0.866185
Train Mean AUC 0.863062 ± 0.002614
Val AUC for fold 0 : 0.860850
Val AUC for fold 1 : 0.857256
Val Mean AUC 0.858573 ± 0.001797
Random Guess
Train Around the mean TPRs
Val Around the mean TPRs

ROC - ROC Plot: Train & Validation of Ensemble (max_depth:3, num_estimators:500)
True Positive Rates
False Positive Rates
0.0
0.2
0.4
0.6
0.8
1.0
Train AUC for fold 0 : 0.863998
Train AUC for fold 1 : 0.868317
Train Mean AUC 0.865645 ± 0.002159
Val AUC for fold 0 : 0.863130
Val AUC for fold 1 : 0.859314
Val Mean AUC 0.860779 ± 0.001908
Random Guess
Train Around the mean TPRs
Val Around the mean TPRs

ROC - ROC Plot: Train & Validation of Ensemble (max_depth:5, num_estimators:20)
True Positive Rates
False Positive Rates
0.0
0.2
0.4
0.6
0.8
1.0
Train AUC for fold 0 : 0.846864
Train AUC for fold 1 : 0.851511
Train Mean AUC 0.848741 ± 0.002323
Val AUC for fold 0 : 0.841839
Val AUC for fold 1 : 0.837390
Val Mean AUC 0.839224 ± 0.002225
Random Guess
Train Around the mean TPRs
Val Around the mean TPRs

ROC - ROC Plot: Train & Validation of Ensemble (max_depth:5, num_estimators:60)
True Positive Rates
False Positive Rates
0.0
0.2
0.4
0.6
0.8
1.0
Train AUC for fold 0 : 0.852383
Train AUC for fold 1 : 0.875185
Train Mean AUC 0.863248 ± 0.011401
Val AUC for fold 0 : 0.846361
Val AUC for fold 1 : 0.860390
Val Mean AUC 0.852920 ± 0.007014
Random Guess
Train Around the mean TPRs
Val Around the mean TPRs

ROC - ROC Plot: Train & Validation of Ensemble (max_depth:5, num_estimators:120)
True Positive Rates
False Positive Rates
0.0
0.2
0.4
0.6
0.8
1.0
Train AUC for fold 0 : 0.871769
Train AUC for fold 1 : 0.875478
Train Mean AUC 0.873096 ± 0.001854
Val AUC for fold 0 : 0.866086
Val AUC for fold 1 : 0.861955
Val Mean AUC 0.863595 ± 0.002066
Random Guess
Train Around the mean TPRs
Val Around the mean TPRs

ROC - ROC Plot: Train & Validation of Ensemble (max_depth:5, num_estimators:300)
True Positive Rates
False Positive Rates
0.0
0.2
0.4
0.6
0.8
1.0
Train AUC for fold 0 : 0.875221
Train AUC for fold 1 : 0.882660
Train Mean AUC 0.878359 ± 0.003719
Val AUC for fold 0 : 0.869462
Val AUC for fold 1 : 0.868558
Val Mean AUC 0.868502 ± 0.000452
Random Guess
Train Around the mean TPRs
Val Around the mean TPRs

ROC - ROC Plot: Train & Validation of Ensemble (max_depth:5, num_estimators:500)
True Positive Rates
False Positive Rates
0.0
0.2
0.4
0.6
0.8
1.0
Train AUC for fold 0 : 0.873151
Train AUC for fold 1 : 0.880660
Train Mean AUC 0.876294 ± 0.003755
Val AUC for fold 0 : 0.868278
Val AUC for fold 1 : 0.866400
Val Mean AUC 0.866791 ± 0.000939
Random Guess
Train Around the mean TPRs
Val Around the mean TPRs

ROC - ROC Plot: Train & Validation of Ensemble (max_depth:10, num_estimators:20)
True Positive Rates
False Positive Rates
0.0
0.2
0.4
0.6
0.8
1.0
Train AUC for fold 0 : 0.896585
Train AUC for fold 1 : 0.898156
Train Mean AUC 0.896723 ± 0.000785
Val AUC for fold 0 : 0.870771
Val AUC for fold 1 : 0.865261
Val Mean AUC 0.867650 ± 0.002755
Random Guess
Train Around the mean TPRs
Val Around the mean TPRs

ROC - ROC Plot: Train & Validation of Ensemble (max_depth:10, num_estimators:60)
True Positive Rates
False Positive Rates
0.0
0.2
0.4
0.6
0.8
1.0
Train AUC for fold 0 : 0.905351
Train AUC for fold 1 : 0.908581
Train Mean AUC 0.906177 ± 0.001615
Val AUC for fold 0 : 0.878836
Val AUC for fold 1 : 0.876224
Val Mean AUC 0.877003 ± 0.001306
Random Guess
Train Around the mean TPRs
Val Around the mean TPRs

ROC - ROC Plot: Train & Validation of Ensemble (max_depth:10, num_estimators:120)
True Positive Rates
False Positive Rates
0.0
0.2
0.4
0.6
0.8
1.0
Train AUC for fold 0 : 0.908238
Train AUC for fold 1 : 0.913080
Train Mean AUC 0.909781 ± 0.002421
Val AUC for fold 0 : 0.882527
Val AUC for fold 1 : 0.880097
Val Mean AUC 0.880748 ± 0.001215
Random Guess
Train Around the mean TPRs
Val Around the mean TPRs

ROC - ROC Plot: Train & Validation of Ensemble (max_depth:10, num_estimators:300)
True Positive Rates
False Positive Rates
0.0
0.2
0.4
0.6
0.8
1.0
Train AUC for fold 0 : 0.905203
Train AUC for fold 1 : 0.912921
Train Mean AUC 0.908307 ± 0.003859
Val AUC for fold 0 : 0.880451
Val AUC for fold 1 : 0.880901
Val Mean AUC 0.880130 ± 0.000225
Random Guess
Train Around the mean TPRs
Val Around the mean TPRs

ROC - ROC Plot: Train & Validation of Ensemble (max_depth:10, num_estimators:500)
True Positive Rates
False Positive Rates
0.0
0.2
0.4
0.6
0.8
1.0
Train AUC for fold 0 : 0.909161
Train AUC for fold 1 : 0.914477
Train Mean AUC 0.910980 ± 0.002658
Val AUC for fold 0 : 0.884042
Val AUC for fold 1 : 0.881601
Val Mean AUC 0.882263 ± 0.001220
Random Guess
Train Around the mean TPRs
Val Around the mean TPRs

ROC - ROC Plot: Train & Validation of Ensemble (max_depth:20, num_estimators:20)
True Positive Rates
False Positive Rates
0.0
0.2
0.4
0.6
0.8
1.0
Train AUC for fold 0 : 0.955309
Train AUC for fold 1 : 0.955902
Train Mean AUC 0.954010 ± 0.000296
Val AUC for fold 0 : 0.886102
Val AUC for fold 1 : 0.881037
Val Mean AUC 0.882972 ± 0.002532
Random Guess
Train Around the mean TPRs
Val Around the mean TPRs

ROC - ROC Plot: Train & Validation of Ensemble (max_depth:20, num_estimators:60)
True Positive Rates
False Positive Rates
0.0
0.2
0.4
0.6
0.8
1.0
Train AUC for fold 0 : 0.962914
Train AUC for fold 1 : 0.963168
Train Mean AUC 0.961226 ± 0.000127
Val AUC for fold 0 : 0.898104
Val AUC for fold 1 : 0.894000
Val Mean AUC 0.895407 ± 0.002052
Random Guess
Train Around the mean TPRs
Val Around the mean TPRs

ROC - ROC Plot: Train & Validation of Ensemble (max_depth:20, num_estimators:120)
True Positive Rates
False Positive Rates
0.0
0.2
0.4
0.6
0.8
1.0
Train AUC for fold 0 : 0.965236
Train AUC for fold 1 : 0.963530
Train Mean AUC 0.962561 ± 0.000853
Val AUC for fold 0 : 0.899724
Val AUC for fold 1 : 0.895716
Val Mean AUC 0.897108 ± 0.002004
Random Guess
Train Around the mean TPRs
Val Around the mean TPRs

ROC - ROC Plot: Train & Validation of Ensemble (max_depth:20, num_estimators:300)
True Positive Rates
False Positive Rates
0.0
0.2
0.4
0.6
0.8
1.0
Train AUC for fold 0 : 0.966231
Train AUC for fold 1 : 0.963943
Train Mean AUC 0.963180 ± 0.001144
Val AUC for fold 0 : 0.902104
Val AUC for fold 1 : 0.896105
Val Mean AUC 0.898448 ± 0.003000
Random Guess
Train Around the mean TPRs
Val Around the mean TPRs

ROC - ROC Plot: Train & Validation of Ensemble (max_depth:20, num_estimators:500)
True Positive Rates
False Positive Rates
0.0
0.2
0.4
0.6
0.8
1.0
Train AUC for fold 0 : 0.965958
Train AUC for fold 1 : 0.964881
Train Mean AUC 0.963524 ± 0.000539
Val AUC for fold 0 : 0.902453
Val AUC for fold 1 : 0.898416
Val Mean AUC 0.899736 ± 0.002018
Random Guess
Train Around the mean TPRs
Val Around the mean TPRs

ROC - ROC Plot: Train & Validation of Ensemble (max_depth:50, num_estimators:20)
True Positive Rates
False Positive Rates
0.0
0.2
0.4
0.6
0.8
1.0
Train AUC for fold 0 : 0.998482
Train AUC for fold 1 : 0.998194
Train Mean AUC 0.993824 ± 0.000144
Val AUC for fold 0 : 0.892401
Val AUC for fold 1 : 0.890593
Val Mean AUC 0.891208 ± 0.000904
Random Guess
Train Around the mean TPRs
Val Around the mean TPRs

ROC - ROC Plot: Train & Validation of Ensemble (max_depth:50, num_estimators:60)
True Positive Rates
False Positive Rates
0.0
0.2
0.4
0.6
0.8
1.0
Train AUC for fold 0 : 0.999095
Train AUC for fold 1 : 0.998722
Train Mean AUC 0.994214 ± 0.000186
Val AUC for fold 0 : 0.906470
Val AUC for fold 1 : 0.901034
Val Mean AUC 0.903606 ± 0.002718
Random Guess
Train Around the mean TPRs
Val Around the mean TPRs

ROC - ROC Plot: Train & Validation of Ensemble (max_depth:50, num_estimators:120)
True Positive Rates
False Positive Rates
0.0
0.2
0.4
0.6
0.8
1.0
Train AUC for fold 0 : 0.999200
Train AUC for fold 1 : 0.999106
Train Mean AUC 0.994407 ± 0.000047
Val AUC for fold 0 : 0.908439
Val AUC for fold 1 : 0.903852
Val Mean AUC 0.905821 ± 0.002294
Random Guess
Train Around the mean TPRs
Val Around the mean TPRs

ROC - ROC Plot: Train & Validation of Ensemble (max_depth:50, num_estimators:300)
True Positive Rates
False Positive Rates
0.0
0.2
0.4
0.6
0.8
1.0
Train AUC for fold 0 : 0.999286
Train AUC for fold 1 : 0.999082
Train Mean AUC 0.994449 ± 0.000102
Val AUC for fold 0 : 0.910791
Val AUC for fold 1 : 0.907138
Val Mean AUC 0.908728 ± 0.001826
Random Guess
Train Around the mean TPRs
Val Around the mean TPRs

ROC - ROC Plot: Train & Validation of Ensemble (max_depth:50, num_estimators:500)
True Positive Rates
False Positive Rates
0.0
0.2
0.4
0.6
0.8
1.0
Train AUC for fold 0 : 0.999268
Train AUC for fold 1 : 0.999101
Train Mean AUC 0.994432 ± 0.000083
Val AUC for fold 0 : 0.911543
Val AUC for fold 1 : 0.907145
Val Mean AUC 0.909093 ± 0.002199
Random Guess
Train Around the mean TPRs
Val Around the mean TPRs

==========
++++++

depth vs AUC
Train AUC
Validation AUC
AUC Scores
depth
1.00
0.98
0.96
0.94
0.92
0.90
0.88
0.86
0.84
10
20
30
40
50

++++++

num_estimators vs AUC
AUC Scores
1.00
0.98
0.96
0.94
0.92
0.90
0.88
0.86
0.84
0
100
200
300
400
500
num_estimatores
Train AUC
Validation AUC

++++++

```
Best hyperparam value:  (50, 500)
```

ROC Curve - Final Model DT (depth:50, n_estimators:500)
True Positive Rates
False Positive Rates
0.0
0.2
0.4
0.6
0.8
1.0
Test AUC for fold 0 : 0.998039
Random Guess
Test Mean AUC 0.993610 ± 0.000000
Test Around the mean TPRs

	Negative	Positive
Precision	0.971022	0.985123
Recall	0.985377	0.970526
Fscore	0.978147	0.977770
Support	25029.000000	24971.000000

```
Test auc score  0.9171612638262937
```

```
              Negative      Positive
Precision     0.492265      0.967025
Recall        0.868678      0.811259
Fscore        0.628417      0.882320
Support    3480.000000  16520.000000
Results Summary:
 [('Hyper Param', (50, 500)), ('AUC', '0.9172'), ('f-score(-ve)', '62.8417'), ('f-score(+ve)', '
```

4.2.2 [A.2] Wordcloud of top 20 important features from SET 1

```
In [12]: # get feature and its importance as tuple
         feature_imp_info = list(zip(feature_name_list, model[0].feature_importances_))

         # filter only those features which have a value greater than zero
         feature_imp_dict = dict(list(filter(lambda x: x[1] > 0.0, feature_imp_info)))

         # create word cloud object for displaying the output
         wc = WordCloud(background_color='white', width=800, height=800)
         wc_output = wc.generate_from_frequencies(feature_imp_dict)
```

```
In [13]: plt.figure(figsize=(8,8))
         plt.imshow(wc_output)
         plt.axis('off')
         plt.tight_layout(pad=0.0)
         plt.title('RF Feature Importances')
         plt.show()
```

4.3 Observation

best hyper param identified is max_depth = 50, and n_estimators=500

Review length is identified as one of the important feature

Positive words such as 'great', 'best' & negative words such as 'disappoint', 'not buy' are recognized as important features by random forest

4.3.1 [A.3] Applying Random Forests on TFIDF, SET 2

In [14]:
```
# form two lists
depth_list = [3, 5, 10, 20, 50] # depends on size of dataset
n_estimators_list = [20, 60, 120, 300, 500] # depends on size of dataset

# create a configuartion dictionary
config_dict = {
    'train_csv_path' : '/home/amd_3/AAIC/ASM_REPO/Processed_data/AMZN_FOOD_REVIW/TFIDF/
    'test_csv_path' : '/home/amd_3/AAIC/ASM_REPO/Processed_data/AMZN_FOOD_REVIW/TFIDF/t
    'train_size' : 50000,
    'test_size' : 20000,
    'hyperparam_list' : list(product(depth_list, n_estimators_list)),
    'implementation': 'rf' # 'xgb' or 'rf'
}
```

In [15]:
```
# read the train, test data and preprocess it
train_features, train_labels, test_features, test_labels = preprocess_data(config_dict,
                                                                           scaling=True
                                                                           dim_reductio

# create a list containing all the features
feature_name_list = train_features.columns.values.tolist()

# train and validate the model
model = train_and_validate_model(config_dict, train_features, train_labels)

# test and evaluate the model
ptabe_entry_a2 = test_and_evaluate_model(config_dict, model, test_features, test_labels
```

```
Train df shape (50000, 503)
Class label distribution in train df:
 0    25029
1    24971
Name: Label, dtype: int64
Test df shape (20000, 503)
Class label distribution in test df:
 1    16520
0     3480
Name: Label, dtype: int64
Shape of ->  train features :50000,501, test features: 20000,501
Shape of -> train labels :50000, test labels: 20000
==========================================================================================
```

ROC - ROC Plot: Train & Validation of Ensemble (max_depth:3, num_estimators:20)
True Positive Rates
False Positive Rates
0.0
0.2
0.4
0.6
0.8
1.0
Train AUC for fold 0 : 0.827722
Train AUC for fold 1 : 0.813675
Train Mean AUC 0.820338 ± 0.007023
Val AUC for fold 0 : 0.821988
Val AUC for fold 1 : 0.806628
Val Mean AUC 0.814011 ± 0.007680
Random Guess
Train Around the mean TPRs
Val Around the mean TPRs

ROC - ROC Plot: Train & Validation of Ensemble (max_depth:3, num_estimators:60)
True Positive Rates
False Positive Rates
0.0
0.2
0.4
0.6
0.8
1.0
Train AUC for fold 0 : 0.849251
Train AUC for fold 1 : 0.849568
Train Mean AUC 0.848892 ± 0.000158
Val AUC for fold 0 : 0.845257
Val AUC for fold 1 : 0.837999
Val Mean AUC 0.841205 ± 0.003629
Random Guess
Train Around the mean TPRs
Val Around the mean TPRs

ROC - ROC Plot: Train & Validation of Ensemble (max_depth:3, num_estimators:120)
True Positive Rates
False Positive Rates
0.0
0.2
0.4
0.6
0.8
1.0
Train AUC for fold 0 : 0.856904
Train AUC for fold 1 : 0.861412
Train Mean AUC 0.858643 ± 0.002254
Val AUC for fold 0 : 0.852558
Val AUC for fold 1 : 0.850922
Val Mean AUC 0.851329 ± 0.000818
Random Guess
Train Around the mean TPRs
Val Around the mean TPRs

ROC - ROC Plot: Train & Validation of Ensemble (max_depth:3, num_estimators:300)
True Positive Rates
False Positive Rates
0.0
0.2
0.4
0.6
0.8
1.0
Train AUC for fold 0 : 0.859956
Train AUC for fold 1 : 0.868039
Train Mean AUC 0.863510 ± 0.004041
Val AUC for fold 0 : 0.855187
Val AUC for fold 1 : 0.857007
Val Mean AUC 0.855655 ± 0.000910
Random Guess
Train Around the mean TPRs
Val Around the mean TPRs

ROC - ROC Plot: Train & Validation of Ensemble (max_depth:3, num_estimators:500)
True Positive Rates
False Positive Rates
0.0
0.2
0.4
0.6
0.8
1.0
Train AUC for fold 0 : 0.860983
Train AUC for fold 1 : 0.871589
Train Mean AUC 0.865736 ± 0.005303
Val AUC for fold 0 : 0.855946
Val AUC for fold 1 : 0.859636
Val Mean AUC 0.857306 ± 0.001845
Random Guess
Train Around the mean TPRs
Val Around the mean TPRs

ROC - ROC Plot: Train & Validation of Ensemble (max_depth:5, num_estimators:20)
True Positive Rates
False Positive Rates
0.0
0.2
0.4
0.6
0.8
1.0
Train AUC for fold 0 : 0.844614
Train AUC for fold 1 : 0.854083
Train Mean AUC 0.848929 ± 0.004735
Val AUC for fold 0 : 0.835381
Val AUC for fold 1 : 0.838155
Val Mean AUC 0.836378 ± 0.001387
Random Guess
Train Around the mean TPRs
Val Around the mean TPRs

ROC - ROC Plot: Train & Validation of Ensemble (max_depth:5, num_estimators:60)
True Positive Rates
False Positive Rates
0.0
0.2
0.4
0.6
0.8
1.0
Train AUC for fold 0 : 0.877316
Train AUC for fold 1 : 0.877880
Train Mean AUC 0.877012 ± 0.000282
Val AUC for fold 0 : 0.867685
Val AUC for fold 1 : 0.860412
Val Mean AUC 0.863644 ± 0.003636
Random Guess
Train Around the mean TPRs
Val Around the mean TPRs

ROC - ROC Plot: Train & Validation of Ensemble (max_depth:5, num_estimators:120)
True Positive Rates
False Positive Rates
0.0
0.2
0.4
0.6
0.8
1.0
Train AUC for fold 0 : 0.867397
Train AUC for fold 1 : 0.880075
Train Mean AUC 0.873100 ± 0.006339
Val AUC for fold 0 : 0.857060
Val AUC for fold 1 : 0.863423
Val Mean AUC 0.859702 ± 0.003182
Random Guess
Train Around the mean TPRs
Val Around the mean TPRs

ROC - ROC Plot: Train & Validation of Ensemble (max_depth:5, num_estimators:300)
True Positive Rates
False Positive Rates
0.0
0.2
0.4
0.6
0.8
1.0
Train AUC for fold 0 : 0.874374
Train AUC for fold 1 : 0.880815
Train Mean AUC 0.876972 ± 0.003220
Val AUC for fold 0 : 0.864041
Val AUC for fold 1 : 0.863499
Val Mean AUC 0.863277 ± 0.000271
Random Guess
Train Around the mean TPRs
Val Around the mean TPRs

ROC - ROC Plot: Train & Validation of Ensemble (max_depth:5, num_estimators:500)
True Positive Rates
False Positive Rates
0.0
0.2
0.4
0.6
0.8
1.0
Train AUC for fold 0 : 0.879351
Train AUC for fold 1 : 0.882900
Train Mean AUC 0.880439 ± 0.001774
Val AUC for fold 0 : 0.869540
Val AUC for fold 1 : 0.865631
Val Mean AUC 0.867063 ± 0.001955
Random Guess
Train Around the mean TPRs
Val Around the mean TPRs

ROC - ROC Plot: Train & Validation of Ensemble (max_depth:10, num_estimators:20)
True Positive Rates
False Positive Rates
0.0
0.2
0.4
0.6
0.8
1.0
Train AUC for fold 0 : 0.899696
Train AUC for fold 1 : 0.899053
Train Mean AUC 0.898655 ± 0.000321
Val AUC for fold 0 : 0.868083
Val AUC for fold 1 : 0.862761
Val Mean AUC 0.864890 ± 0.002661
Random Guess
Train Around the mean TPRs
Val Around the mean TPRs

ROC - ROC Plot: Train & Validation of Ensemble (max_depth:10, num_estimators:60)
True Positive Rates
False Positive Rates
0.0
0.2
0.4
0.6
0.8
1.0
Train AUC for fold 0 : 0.906098
Train AUC for fold 1 : 0.911485
Train Mean AUC 0.907971 ± 0.002694
Val AUC for fold 0 : 0.876292
Val AUC for fold 1 : 0.875683
Val Mean AUC 0.875479 ± 0.000305
Random Guess
Train Around the mean TPRs
Val Around the mean TPRs

ROC - ROC Plot: Train & Validation of Ensemble (max_depth:10, num_estimators:120)
True Positive Rates
False Positive Rates
0.0
0.2
0.4
0.6
0.8
1.0
Train AUC for fold 0 : 0.911484
Train AUC for fold 1 : 0.919052
Train Mean AUC 0.914268 ± 0.003784
Val AUC for fold 0 : 0.880496
Val AUC for fold 1 : 0.880902
Val Mean AUC 0.880109 ± 0.000203
Random Guess
Train Around the mean TPRs
Val Around the mean TPRs

ROC - ROC Plot: Train & Validation of Ensemble (max_depth:10, num_estimators:300)
True Positive Rates
False Positive Rates
0.0
0.2
0.4
0.6
0.8
1.0
Train AUC for fold 0 : 0.916305
Train AUC for fold 1 : 0.917592
Train Mean AUC 0.915999 ± 0.000644
Val AUC for fold 0 : 0.885752
Val AUC for fold 1 : 0.880776
Val Mean AUC 0.882634 ± 0.002488
Random Guess
Train Around the mean TPRs
Val Around the mean TPRs

ROC - ROC Plot: Train & Validation of Ensemble (max_depth:10, num_estimators:500)
True Positive Rates
False Positive Rates
0.0
0.2
0.4
0.6
0.8
1.0
Train AUC for fold 0 : 0.915428
Train AUC for fold 1 : 0.917548
Train Mean AUC 0.915528 ± 0.001060
Val AUC for fold 0 : 0.884243
Val AUC for fold 1 : 0.880935
Val Mean AUC 0.881994 ± 0.001654
Random Guess
Train Around the mean TPRs
Val Around the mean TPRs

ROC - ROC Plot: Train & Validation of Ensemble (max_depth:20, num_estimators:20)
True Positive Rates
False Positive Rates
0.0
0.2
0.4
0.6
0.8
1.0
Train AUC for fold 0 : 0.963298
Train AUC for fold 1 : 0.962311
Train Mean AUC 0.960913 ± 0.000494
Val AUC for fold 0 : 0.889008
Val AUC for fold 1 : 0.883379
Val Mean AUC 0.885705 ± 0.002814
Random Guess
Train Around the mean TPRs
Val Around the mean TPRs

ROC - ROC Plot: Train & Validation of Ensemble (max_depth:20, num_estimators:60)
True Positive Rates
False Positive Rates
0.0
0.2
0.4
0.6
0.8
1.0
Train AUC for fold 0 : 0.966741
Train AUC for fold 1 : 0.968544
Train Mean AUC 0.965444 ± 0.000901
Val AUC for fold 0 : 0.896862
Val AUC for fold 1 : 0.893846
Val Mean AUC 0.894773 ± 0.001508
Random Guess
Train Around the mean TPRs
Val Around the mean TPRs

ROC - ROC Plot: Train & Validation of Ensemble (max_depth:20, num_estimators:120)
True Positive Rates
False Positive Rates
0.0
0.2
0.4
0.6
0.8
1.0
Train AUC for fold 0 : 0.970304
Train AUC for fold 1 : 0.970197
Train Mean AUC 0.967962 ± 0.000053
Val AUC for fold 0 : 0.900855
Val AUC for fold 1 : 0.897395
Val Mean AUC 0.898468 ± 0.001730
Random Guess
Train Around the mean TPRs
Val Around the mean TPRs

ROC - ROC Plot: Train & Validation of Ensemble (max_depth:20, num_estimators:300)
True Positive Rates
False Positive Rates
0.0
0.2
0.4
0.6
0.8
1.0
Train AUC for fold 0 : 0.970489
Train AUC for fold 1 : 0.970966
Train Mean AUC 0.968498 ± 0.000239
Val AUC for fold 0 : 0.901750
Val AUC for fold 1 : 0.898329
Val Mean AUC 0.899424 ± 0.001710
Random Guess
Train Around the mean TPRs
Val Around the mean TPRs

ROC - ROC Plot: Train & Validation of Ensemble (max_depth:20, num_estimators:500)
True Positive Rates
False Positive Rates
0.0
0.2
0.4
0.6
0.8
1.0
Train AUC for fold 0 : 0.970305
Train AUC for fold 1 : 0.971336
Train Mean AUC 0.968490 ± 0.000516
Val AUC for fold 0 : 0.901379
Val AUC for fold 1 : 0.898980
Val Mean AUC 0.899525 ± 0.001200
Random Guess
Train Around the mean TPRs
Val Around the mean TPRs

ROC - ROC Plot: Train & Validation of Ensemble (max_depth:50, num_estimators:20)
True Positive Rates
False Positive Rates
0.0
0.2
0.4
0.6
0.8
1.0
Train AUC for fold 0 : 0.999035
Train AUC for fold 1 : 0.999110
Train Mean AUC 0.994295 ± 0.000037
Val AUC for fold 0 : 0.891056
Val AUC for fold 1 : 0.888806
Val Mean AUC 0.889828 ± 0.001125
Random Guess
Train Around the mean TPRs
Val Around the mean TPRs

ROC - ROC Plot: Train & Validation of Ensemble (max_depth:50, num_estimators:60)
True Positive Rates
False Positive Rates
0.0
0.2
0.4
0.6
0.8
1.0
Train AUC for fold 0 : 0.999571
Train AUC for fold 1 : 0.999535
Train Mean AUC 0.994648 ± 0.000018
Val AUC for fold 0 : 0.902839
Val AUC for fold 1 : 0.894534
Val Mean AUC 0.898584 ± 0.004152
Random Guess
Train Around the mean TPRs
Val Around the mean TPRs

ROC - ROC Plot: Train & Validation of Ensemble (max_depth:50, num_estimators:120)
True Positive Rates
False Positive Rates
0.0
0.2
0.4
0.6
0.8
1.0
Train AUC for fold 0 : 0.999668
Train AUC for fold 1 : 0.999693
Train Mean AUC 0.994730 ± 0.000013
Val AUC for fold 0 : 0.904133
Val AUC for fold 1 : 0.897734
Val Mean AUC 0.900865 ± 0.003200
Random Guess
Train Around the mean TPRs
Val Around the mean TPRs

ROC - ROC Plot: Train & Validation of Ensemble (max_depth:50, num_estimators:300)
True Positive Rates
False Positive Rates
0.0
0.2
0.4
0.6
0.8
1.0
Train AUC for fold 0 : 0.999708
Train AUC for fold 1 : 0.999691
Train Mean AUC 0.994751 ± 0.000009
Val AUC for fold 0 : 0.907649
Val AUC for fold 1 : 0.897764
Val Mean AUC 0.902601 ± 0.004942
Random Guess
Train Around the mean TPRs
Val Around the mean TPRs

```
==========================================================================================
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++


 The depth vs AUC score plot
```

depth vs AUC
1.00
0.95
0.90
0.85
AUC Scores
10
20
30
40
50
depth
Train AUC
Validation AUC

```
++++++
```

num_estimators vs AUC
1.00
0.95
0.90
0.85
AUC Scores
0
100
200
300
400
500
num_estimatores
Train AUC
Validation AUC

++++++

Hyper param Heatmap DT-Train
max_depth
3
5
10
20
50
0.82 0.85 0.86 0.86 0.87
0.85 0.88 0.87 0.88 0.88
0.9 0.91 0.91 0.92 0.92
0.96 0.97 0.97 0.97 0.97
0.99 0.99 0.99 0.99 0.99
20 60 120 300 500
n_estimators
0.99
0.96
0.93
0.90
0.87
0.84

Hyper param Heatmap DT-Validation
max_depth
3
5
10
20
50
0.81 0.84 0.85 0.86 0.86
0.84 0.86 0.86 0.86 0.87
0.86 0.88 0.88 0.88 0.88
0.89 0.89 0.9 0.9 0.9
0.89 0.9 0.9 0.9 0.9
20 60 120 300 500
n_estimators
0.900
0.885
0.870
0.855
0.840
0.825

```
Best hyperparam value:  (50, 300)
```

	Negative	Positive
Precision	0.975360	0.988269
Recall	0.988453	0.974971
Fscore	0.981863	0.981575
Support	25029.000000	24971.000000

```
Test auc score  0.9147983054415931
```

```
              Negative      Positive
Precision     0.493032      0.965974
Recall        0.864080      0.812833
Fscore        0.627832      0.882811
Support    3480.000000  16520.000000
Results Summary:
 [('Hyper Param', (50, 300)), ('AUC', '0.9148'), ('f-score(-ve)', '62.7832'), ('f-score(+ve)', '
```

4.3.2 [A.4] Wordcloud of top 20 important features from SET 2

In [16]:
```
# get feature and its importance as tuple
feature_imp_info = list(zip(feature_name_list, model[0].feature_importances_))

# filter only those features which have a value greater than zero
feature_imp_dict = dict(list(filter(lambda x: x[1] > 0.0, feature_imp_info)))

# create word cloud object for displaying the output
wc = WordCloud(background_color='white', width=800, height=800)
wc_output = wc.generate_from_frequencies(feature_imp_dict)
```

```
In [17]: plt.figure(figsize=(8,8))
         plt.imshow(wc_output)
         plt.axis('off')
         plt.tight_layout(pad=0.0)
         plt.title('RF Feature Importances')
         plt.show()
```

4.4 Observation

best hyper param identified is max_depth = 50, and n_estimators=300

Review length is identified as one of the important feature

Positive words such as 'great', 'best' & negative words such as 'disappoint', 'bad' are recognized as important features by random forest

4.4.1 [A.5] Applying Random Forests on AVG W2V, SET 3

In [18]:
```
# form two lists
depth_list = [3, 5, 10, 20, 50] # depends on size of dataset
n_estimators_list = [20, 60, 120, 300, 500] # depends on size of dataset

# create a configuartion dictionary
config_dict = {
    'train_csv_path' : '/home/amd_3/AAIC/ASM_REPO/Processed_data/AMZN_FOOD_REVIW/AVG_W2
    'test_csv_path' : '/home/amd_3/AAIC/ASM_REPO/Processed_data/AMZN_FOOD_REVIW/AVG_W2V
    'train_size' : 50000,
    'test_size' : 20000,
    'hyperparam_list' : list(product(depth_list, n_estimators_list)),
    'implementation': 'rf' # 'xgb' or 'rf'
}
```

In [19]:
```
# read the train, test data and preprocess it
train_features, train_labels, test_features, test_labels = preprocess_data(config_dict,
                                                                            scaling=True
                                                                            dim_reductio

# train and validate the model
model = train_and_validate_model(config_dict, train_features, train_labels)

# test and evaluate the model
ptabe_entry_a3 = test_and_evaluate_model(config_dict, model, test_features, test_labels
```

```
Train df shape (50000, 52)
Class label distribution in train df:
 0    25029
1    24971
Name: Label, dtype: int64
Test df shape (20000, 52)
Class label distribution in test df:
 1    16520
0     3480
Name: Label, dtype: int64
Shape of ->  train features :50000,50, test features: 20000,50
Shape of -> train labels :50000, test labels: 20000
==========================================================================================
```

ROC - ROC Plot: Train & Validation of Ensemble (max_depth:3, num_estimators:20)
True Positive Rates
False Positive Rates
0.0
0.2
0.4
0.6
0.8
1.0
Train AUC for fold 0 : 0.871207
Train AUC for fold 1 : 0.864396
Train Mean AUC 0.867427 ± 0.003405
Val AUC for fold 0 : 0.865999
Val AUC for fold 1 : 0.856274
Val Mean AUC 0.860817 ± 0.004863
Random Guess
Train Around the mean TPRs
Val Around the mean TPRs

ROC - ROC Plot: Train & Validation of Ensemble (max_depth:3, num_estimators:60)
True Positive Rates
False Positive Rates
0.0
0.2
0.4
0.6
0.8
1.0
Train AUC for fold 0 : 0.870880
Train AUC for fold 1 : 0.875354
Train Mean AUC 0.872597 ± 0.002237
Val AUC for fold 0 : 0.865432
Val AUC for fold 1 : 0.867304
Val Mean AUC 0.865943 ± 0.000936
Random Guess
Train Around the mean TPRs
Val Around the mean TPRs

ROC - ROC Plot: Train & Validation of Ensemble (max_depth:3, num_estimators:120)
True Positive Rates
False Positive Rates
0.0
0.2
0.4
0.6
0.8
1.0
Train AUC for fold 0 : 0.878304
Train AUC for fold 1 : 0.878625
Train Mean AUC 0.877967 ± 0.000160
Val AUC for fold 0 : 0.872435
Val AUC for fold 1 : 0.870545
Val Mean AUC 0.871073 ± 0.000945
Random Guess
Train Around the mean TPRs
Val Around the mean TPRs

ROC - ROC Plot: Train & Validation of Ensemble (max_depth:3, num_estimators:300)
True Positive Rates
False Positive Rates
0.0
0.2
0.4
0.6
0.8
1.0
Train AUC for fold 0 : 0.881333
Train AUC for fold 1 : 0.876547
Train Mean AUC 0.878449 ± 0.002393
Val AUC for fold 0 : 0.875343
Val AUC for fold 1 : 0.869290
Val Mean AUC 0.871871 ± 0.003027
Random Guess
Train Around the mean TPRs
Val Around the mean TPRs

ROC - ROC Plot: Train & Validation of Ensemble (max_depth:3, num_estimators:500)
True Positive Rates
False Positive Rates
0.0
0.2
0.4
0.6
0.8
1.0
Train AUC for fold 0 : 0.877822
Train AUC for fold 1 : 0.880259
Train Mean AUC 0.878557 ± 0.001219
Val AUC for fold 0 : 0.871836
Val AUC for fold 1 : 0.872457
Val Mean AUC 0.871734 ± 0.000311
Random Guess
Train Around the mean TPRs
Val Around the mean TPRs

ROC - ROC Plot: Train & Validation of Ensemble (max_depth:5, num_estimators:20)
True Positive Rates
False Positive Rates
0.0
0.2
0.4
0.6
0.8
1.0
Train AUC for fold 0 : 0.893827
Train AUC for fold 1 : 0.894513
Train Mean AUC 0.893547 ± 0.000343
Val AUC for fold 0 : 0.881157
Val AUC for fold 1 : 0.880655
Val Mean AUC 0.880454 ± 0.000251
Random Guess
Train Around the mean TPRs
Val Around the mean TPRs

ROC - ROC Plot: Train & Validation of Ensemble (max_depth:5, num_estimators:60)
True Positive Rates
False Positive Rates
0.0
0.2
0.4
0.6
0.8
1.0
Train AUC for fold 0 : 0.896955
Train AUC for fold 1 : 0.899230
Train Mean AUC 0.897477 ± 0.001137
Val AUC for fold 0 : 0.884987
Val AUC for fold 1 : 0.885790
Val Mean AUC 0.884917 ± 0.000401
Random Guess
Train Around the mean TPRs
Val Around the mean TPRs

ROC - ROC Plot: Train & Validation of Ensemble (max_depth:5, num_estimators:120)
True Positive Rates
False Positive Rates
0.0
0.2
0.4
0.6
0.8
1.0
Train AUC for fold 0 : 0.897574
Train AUC for fold 1 : 0.898492
Train Mean AUC 0.897383 ± 0.000459
Val AUC for fold 0 : 0.885108
Val AUC for fold 1 : 0.885446
Val Mean AUC 0.884746 ± 0.000169
Random Guess
Train Around the mean TPRs
Val Around the mean TPRs

ROC - ROC Plot: Train & Validation of Ensemble (max_depth:5, num_estimators:300)
True Positive Rates
False Positive Rates
0.0
0.2
0.4
0.6
0.8
1.0
Train AUC for fold 0 : 0.900041
Train AUC for fold 1 : 0.900674
Train Mean AUC 0.899701 ± 0.000316
Val AUC for fold 0 : 0.887101
Val AUC for fold 1 : 0.887092
Val Mean AUC 0.886601 ± 0.000004
Random Guess
Train Around the mean TPRs
Val Around the mean TPRs

ROC - ROC Plot: Train & Validation of Ensemble (max_depth:5, num_estimators:500)
True Positive Rates
False Positive Rates
0.0
0.2
0.4
0.6
0.8
1.0
Train AUC for fold 0 : 0.899068
Train AUC for fold 1 : 0.900675
Train Mean AUC 0.899193 ± 0.000803
Val AUC for fold 0 : 0.886575
Val AUC for fold 1 : 0.887296
Val Mean AUC 0.886441 ± 0.000361
Random Guess
Train Around the mean TPRs
Val Around the mean TPRs

ROC - ROC Plot: Train & Validation of Ensemble (max_depth:10, num_estimators:20)
True Positive Rates
False Positive Rates
0.0
0.2
0.4
0.6
0.8
1.0
Train AUC for fold 0 : 0.968710
Train AUC for fold 1 : 0.971357
Train Mean AUC 0.967911 ± 0.001323
Val AUC for fold 0 : 0.898840
Val AUC for fold 1 : 0.898574
Val Mean AUC 0.898087 ± 0.000133
Random Guess
Train Around the mean TPRs
Val Around the mean TPRs

ROC - ROC Plot: Train & Validation of Ensemble (max_depth:10, num_estimators:60)
True Positive Rates
False Positive Rates
0.0
0.2
0.4
0.6
0.8
1.0
Train AUC for fold 0 : 0.973537
Train AUC for fold 1 : 0.974758
Train Mean AUC 0.971730 ± 0.000611
Val AUC for fold 0 : 0.904959
Val AUC for fold 1 : 0.905214
Val Mean AUC 0.904413 ± 0.000127
Random Guess
Train Around the mean TPRs
Val Around the mean TPRs

ROC - ROC Plot: Train & Validation of Ensemble (max_depth:10, num_estimators:120)
True Positive Rates
False Positive Rates
0.0
0.2
0.4
0.6
0.8
1.0
Train AUC for fold 0 : 0.974541
Train AUC for fold 1 : 0.975638
Train Mean AUC 0.972568 ± 0.000548
Val AUC for fold 0 : 0.906885
Val AUC for fold 1 : 0.906945
Val Mean AUC 0.906218 ± 0.000030
Random Guess
Train Around the mean TPRs
Val Around the mean TPRs

ROC - ROC Plot: Train & Validation of Ensemble (max_depth:10, num_estimators:300)
True Positive Rates
False Positive Rates
0.0
0.2
0.4
0.6
0.8
1.0
Train AUC for fold 0 : 0.974949
Train AUC for fold 1 : 0.976161
Train Mean AUC 0.972966 ± 0.000606
Val AUC for fold 0 : 0.907244
Val AUC for fold 1 : 0.907757
Val Mean AUC 0.906807 ± 0.000257
Random Guess
Train Around the mean TPRs
Val Around the mean TPRs

ROC - ROC Plot: Train & Validation of Ensemble (max_depth:10, num_estimators:500)
True Positive Rates
False Positive Rates
0.0
0.2
0.4
0.6
0.8
1.0
Train AUC for fold 0 : 0.975374
Train AUC for fold 1 : 0.976373
Train Mean AUC 0.973289 ± 0.000500
Val AUC for fold 0 : 0.907428
Val AUC for fold 1 : 0.907891
Val Mean AUC 0.906998 ± 0.000231
Random Guess
Train Around the mean TPRs
Val Around the mean TPRs

ROC - ROC Plot: Train & Validation of Ensemble (max_depth:20, num_estimators:20)
True Positive Rates
False Positive Rates
0.0
0.2
0.4
0.6
0.8
1.0
Train AUC for fold 0 : 0.999981
Train AUC for fold 1 : 0.999978
Train Mean AUC 0.994944 ± 0.000001
Val AUC for fold 0 : 0.873769
Val AUC for fold 1 : 0.881548
Val Mean AUC 0.877558 ± 0.003890
Random Guess
Train Around the mean TPRs
Val Around the mean TPRs

ROC - ROC Plot: Train & Validation of Ensemble (max_depth:20, num_estimators:60)
True Positive Rates
False Positive Rates
0.0
0.2
0.4
0.6
0.8
1.0
Train AUC for fold 0 : 1.000000
Train AUC for fold 1 : 1.000000
Train Mean AUC 0.994949 ± 0.000000
Val AUC for fold 0 : 0.863104
Val AUC for fold 1 : 0.837154
Val Mean AUC 0.850103 ± 0.012975
Random Guess
Train Around the mean TPRs
Val Around the mean TPRs

ROC - ROC Plot: Train & Validation of Ensemble (max_depth:20, num_estimators:120)
True Positive Rates
False Positive Rates
0.0
0.2
0.4
0.6
0.8
1.0
Train AUC for fold 0 : 1.000000
Train AUC for fold 1 : 1.000000
Train Mean AUC 0.994949 ± 0.000000
Val AUC for fold 0 : 0.848835
Val AUC for fold 1 : 0.853499
Val Mean AUC 0.851125 ± 0.002332
Random Guess
Train Around the mean TPRs
Val Around the mean TPRs

ROC - ROC Plot: Train & Validation of Ensemble (max_depth:20, num_estimators:300)
True Positive Rates
False Positive Rates
0.0
0.2
0.4
0.6
0.8
1.0
Train AUC for fold 0 : 1.000000
Train AUC for fold 1 : 1.000000
Train Mean AUC 0.994949 ± 0.000000
Val AUC for fold 0 : 0.848334
Val AUC for fold 1 : 0.860961
Val Mean AUC 0.854599 ± 0.006313
Random Guess
Train Around the mean TPRs
Val Around the mean TPRs

ROC - ROC Plot: Train & Validation of Ensemble (max_depth:20, num_estimators:500)
True Positive Rates
False Positive Rates
0.0
0.2
0.4
0.6
0.8
1.0
Train AUC for fold 0 : 1.000000
Train AUC for fold 1 : 1.000000
Train Mean AUC 0.994949 ± 0.000000
Val AUC for fold 0 : 0.850966
Val AUC for fold 1 : 0.863624
Val Mean AUC 0.857246 ± 0.006329
Random Guess
Train Around the mean TPRs
Val Around the mean TPRs

ROC - ROC Plot: Train & Validation of Ensemble (max_depth:50, num_estimators:20)
True Positive Rates
False Positive Rates
0.0
0.2
0.4
0.6
0.8
1.0
Train AUC for fold 0 : 0.999975
Train AUC for fold 1 : 0.999986
Train Mean AUC 0.994944 ± 0.000006
Val AUC for fold 0 : 0.882016
Val AUC for fold 1 : 0.878231
Val Mean AUC 0.880061 ± 0.001892
Random Guess
Train Around the mean TPRs
Val Around the mean TPRs

ROC - ROC Plot: Train & Validation of Ensemble (max_depth:50, num_estimators:60)
True Positive Rates
False Positive Rates
0.0
0.2
0.4
0.6
0.8
1.0
Train AUC for fold 0 : 1.000000
Train AUC for fold 1 : 1.000000
Train Mean AUC 0.994949 ± 0.000000
Val AUC for fold 0 : 0.841816
Val AUC for fold 1 : 0.833149
Val Mean AUC 0.837433 ± 0.004334
Random Guess
Train Around the mean TPRs
Val Around the mean TPRs

ROC - ROC Plot: Train & Validation of Ensemble (max_depth:50, num_estimators:120)
True Positive Rates
False Positive Rates
0.0
0.2
0.4
0.6
0.8
1.0
Train AUC for fold 0 : 1.000000
Train AUC for fold 1 : 1.000000
Train Mean AUC 0.994949 ± 0.000000
Val AUC for fold 0 : 0.865368
Val AUC for fold 1 : 0.850062
Val Mean AUC 0.857642 ± 0.007653
Random Guess
Train Around the mean TPRs
Val Around the mean TPRs

ROC - ROC Plot: Train & Validation of Ensemble (max_depth:50, num_estimators:300)
True Positive Rates
False Positive Rates
0.0
0.2
0.4
0.6
0.8
1.0
Train AUC for fold 0 : 1.000000
Train AUC for fold 1 : 1.000000
Train Mean AUC 0.994949 ± 0.000000
Val AUC for fold 0 : 0.880163
Val AUC for fold 1 : 0.879637
Val Mean AUC 0.879826 ± 0.000263
Random Guess
Train Around the mean TPRs
Val Around the mean TPRs

```
==========================================================================================
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++


 The depth vs AUC score plot
```

depth vs AUC
1.000
0.975
0.950
0.925
0.900
0.875
0.850
AUC Scores
Train AUC
Validation AUC
10
20
30
40
50
depth

```
++++++
```

num_estimators vs AUC
1.000
0.975
0.950
0.925
0.900
0.875
0.850
AUC Scores
Train AUC
Validation AUC
0
100
200
300
400
500
num_estimatores

++++++

Hyper param Heatmap DT-Train
max_depth
3
5
10
20
50
0.87 0.87 0.88 0.88 0.88
0.89 0.9 0.9 0.9 0.9
0.97 0.97 0.97 0.97 0.97
0.99 0.99 0.99 0.99 0.99
0.99 0.99 0.99 0.99 0.99
20 60 120 300 500
n_estimators
0.975
0.950
0.925
0.900
0.875

Hyper param Heatmap DT-Validation
max_depth
3
5
10
20
50
0.86 0.87 0.87 0.87 0.87
0.88 0.88 0.88 0.89 0.89
0.9 0.9 0.91 0.91 0.91
0.88 0.85 0.85 0.85 0.86
0.88 0.84 0.86 0.88 0.89
20 60 120 300 500
n_estimators
0.900
0.885
0.870
0.855
0.840

```
Best hyperparam value:  (10, 500)
```

	Negative	Positive
Precision	0.885323	0.899515
Recall	0.901594	0.882944
Fscore	0.893385	0.891152
Support	25029.000000	24971.000000

```
Test auc score  0.9112048229359623
```

```
              Negative      Positive
Precision     0.482475      0.964299
Recall        0.858333      0.806053
Fscore        0.617723      0.878103
Support    3480.000000  16520.000000
Results Summary:
 [('Hyper Param', (10, 500)), ('AUC', '0.9112'), ('f-score(-ve)', '61.7723'), ('f-score(+ve)', '
```

best hyper param identified is max_depth = 10, and n_estimators=500

4.4.2 [A.6] Applying Random Forests on TFIDF W2V, SET 4

```
In [20]: # form two lists
         depth_list = [3, 5, 10, 20, 50] # depends on size of dataset
         n_estimators_list = [20, 60, 120, 300, 500] # depends on size of dataset

         # create a configuarion dictionary
         config_dict = {
             'train_csv_path' : '/home/amd_3/AAIC/ASM_REPO/Processed_data/AMZN_FOOD_REVIW/TFIDF_
```

```
            'test_csv_path' : '/home/amd_3/AAIC/ASM_REPO/Processed_data/AMZN_FOOD_REVIW/TFIDF_W
            'train_size' : 50000,
            'test_size' : 20000,
            'hyperparam_list' : list(product(depth_list, n_estimators_list)),
            'implementation': 'rf' # 'xgb' or 'rf'
        }
```

In [21]:
```
# read the train, test data and preprocess it
train_features, train_labels, test_features, test_labels = preprocess_data(config_dict,
                                                                           scaling=True
                                                                           dim_reductio

# train and validate the model
model = train_and_validate_model(config_dict, train_features, train_labels)

# test and evaluate the model
ptabe_entry_a4 = test_and_evaluate_model(config_dict, model, test_features, test_labels
```

```
Train df shape (50000, 52)
Class label distribution in train df:
 0    25029
1    24971
Name: Label, dtype: int64
Test df shape (20000, 52)
Class label distribution in test df:
 1    16520
0     3480
Name: Label, dtype: int64
Shape of ->  train features :50000,50, test features: 20000,50
Shape of -> train labels :50000, test labels: 20000
==========================================================================================
```

ROC - ROC Plot: Train & Validation of Ensemble (max_depth:3, num_estimators:20)
True Positive Rates
False Positive Rates
0.0
0.2
0.4
0.6
0.8
1.0
Train AUC for fold 0 : 0.806717
Train AUC for fold 1 : 0.806535
Train Mean AUC 0.806328 ± 0.000091
Val AUC for fold 0 : 0.799308
Val AUC for fold 1 : 0.791184
Val Mean AUC 0.794960 ± 0.004062
Random Guess
Train Around the mean TPRs
Val Around the mean TPRs

ROC - ROC Plot: Train & Validation of Ensemble (max_depth:3, num_estimators:60)
True Positive Rates
False Positive Rates
0.0
0.2
0.4
0.6
0.8
1.0
Train AUC for fold 0 : 0.818514
Train AUC for fold 1 : 0.815842
Train Mean AUC 0.816838 ± 0.001336
Val AUC for fold 0 : 0.811497
Val AUC for fold 1 : 0.805108
Val Mean AUC 0.808052 ± 0.003195
Random Guess
Train Around the mean TPRs
Val Around the mean TPRs

ROC - ROC Plot: Train & Validation of Ensemble (max_depth:3, num_estimators:120)
True Positive Rates
False Positive Rates
0.0
0.2
0.4
0.6
0.8
1.0
Train AUC for fold 0 : 0.816016
Train AUC for fold 1 : 0.822034
Train Mean AUC 0.818678 ± 0.003009
Val AUC for fold 0 : 0.809866
Val AUC for fold 1 : 0.810551
Val Mean AUC 0.809930 ± 0.000342
Random Guess
Train Around the mean TPRs
Val Around the mean TPRs

ROC - ROC Plot: Train & Validation of Ensemble (max_depth:3, num_estimators:300)
True Positive Rates
False Positive Rates
0.0
0.2
0.4
0.6
0.8
1.0
Train AUC for fold 0 : 0.817483
Train AUC for fold 1 : 0.819922
Train Mean AUC 0.818385 ± 0.001219
Val AUC for fold 0 : 0.810960
Val AUC for fold 1 : 0.808292
Val Mean AUC 0.809355 ± 0.001334
Random Guess
Train Around the mean TPRs
Val Around the mean TPRs

ROC - ROC Plot: Train & Validation of Ensemble (max_depth:3, num_estimators:500)
True Positive Rates
False Positive Rates
0.0
0.2
0.4
0.6
0.8
1.0
Train AUC for fold 0 : 0.820996
Train AUC for fold 1 : 0.823149
Train Mean AUC 0.821753 ± 0.001077
Val AUC for fold 0 : 0.814384
Val AUC for fold 1 : 0.810471
Val Mean AUC 0.812179 ± 0.001957
Random Guess
Train Around the mean TPRs
Val Around the mean TPRs

ROC - ROC Plot: Train & Validation of Ensemble (max_depth:5, num_estimators:20)
True Positive Rates
False Positive Rates
0.0
0.2
0.4
0.6
0.8
1.0
Train AUC for fold 0 : 0.837136
Train AUC for fold 1 : 0.839996
Train Mean AUC 0.838219 ± 0.001430
Val AUC for fold 0 : 0.819158
Val AUC for fold 1 : 0.819228
Val Mean AUC 0.818901 ± 0.000035
Random Guess
Train Around the mean TPRs
Val Around the mean TPRs

ROC - ROC Plot: Train & Validation of Ensemble (max_depth:5, num_estimators:60)
True Positive Rates
False Positive Rates
0.0
0.2
0.4
0.6
0.8
1.0
Train AUC for fold 0 : 0.845137
Train AUC for fold 1 : 0.847454
Train Mean AUC 0.845837 ± 0.001158
Val AUC for fold 0 : 0.828102
Val AUC for fold 1 : 0.826040
Val Mean AUC 0.826772 ± 0.001031
Random Guess
Train Around the mean TPRs
Val Around the mean TPRs

ROC - ROC Plot: Train & Validation of Ensemble (max_depth:5, num_estimators:120)
True Positive Rates
False Positive Rates
0.0
0.2
0.4
0.6
0.8
1.0
Train AUC for fold 0 : 0.846557
Train AUC for fold 1 : 0.848670
Train Mean AUC 0.847185 ± 0.001056
Val AUC for fold 0 : 0.828957
Val AUC for fold 1 : 0.827047
Val Mean AUC 0.827704 ± 0.000955
Random Guess
Train Around the mean TPRs
Val Around the mean TPRs

ROC - ROC Plot: Train & Validation of Ensemble (max_depth:5, num_estimators:300)
True Positive Rates
False Positive Rates
0.0
0.2
0.4
0.6
0.8
1.0
Train AUC for fold 0 : 0.846009
Train AUC for fold 1 : 0.849246
Train Mean AUC 0.847201 ± 0.001618
Val AUC for fold 0 : 0.829462
Val AUC for fold 1 : 0.827783
Val Mean AUC 0.828309 ± 0.000839
Random Guess
Train Around the mean TPRs
Val Around the mean TPRs

ROC - ROC Plot: Train & Validation of Ensemble (max_depth:5, num_estimators:500)
True Positive Rates
False Positive Rates
0.0
0.2
0.4
0.6
0.8
1.0
Train AUC for fold 0 : 0.845693
Train AUC for fold 1 : 0.850024
Train Mean AUC 0.847422 ± 0.002165
Val AUC for fold 0 : 0.829518
Val AUC for fold 1 : 0.827844
Val Mean AUC 0.828363 ± 0.000837
Random Guess
Train Around the mean TPRs
Val Around the mean TPRs

ROC - ROC Plot: Train & Validation of Ensemble (max_depth:10, num_estimators:20)
True Positive Rates
False Positive Rates
0.0
0.2
0.4
0.6
0.8
1.0
Train AUC for fold 0 : 0.950192
Train AUC for fold 1 : 0.952635
Train Mean AUC 0.950143 ± 0.001222
Val AUC for fold 0 : 0.846026
Val AUC for fold 1 : 0.843604
Val Mean AUC 0.844472 ± 0.001211
Random Guess
Train Around the mean TPRs
Val Around the mean TPRs

ROC - ROC Plot: Train & Validation of Ensemble (max_depth:10, num_estimators:60)
True Positive Rates
False Positive Rates
0.0
0.2
0.4
0.6
0.8
1.0
Train AUC for fold 0 : 0.957707
Train AUC for fold 1 : 0.960107
Train Mean AUC 0.957414 ± 0.001200
Val AUC for fold 0 : 0.854759
Val AUC for fold 1 : 0.853467
Val Mean AUC 0.853808 ± 0.000646
Random Guess
Train Around the mean TPRs
Val Around the mean TPRs

ROC - ROC Plot: Train & Validation of Ensemble (max_depth:10, num_estimators:120)
True Positive Rates
False Positive Rates
0.0
0.2
0.4
0.6
0.8
1.0
Train AUC for fold 0 : 0.959895
Train AUC for fold 1 : 0.962229
Train Mean AUC 0.959468 ± 0.001167
Val AUC for fold 0 : 0.856888
Val AUC for fold 1 : 0.854970
Val Mean AUC 0.855570 ± 0.000959
Random Guess
Train Around the mean TPRs
Val Around the mean TPRs

ROC - ROC Plot: Train & Validation of Ensemble (max_depth:10, num_estimators:300)
True Positive Rates
False Positive Rates
0.0
0.2
0.4
0.6
0.8
1.0
Train AUC for fold 0 : 0.960582
Train AUC for fold 1 : 0.963031
Train Mean AUC 0.960182 ± 0.001224
Val AUC for fold 0 : 0.858290
Val AUC for fold 1 : 0.856988
Val Mean AUC 0.857234 ± 0.000651
Random Guess
Train Around the mean TPRs
Val Around the mean TPRs

ROC - ROC Plot: Train & Validation of Ensemble (max_depth:10, num_estimators:500)
True Positive Rates
False Positive Rates
0.0
0.2
0.4
0.6
0.8
1.0
Train AUC for fold 0 : 0.961283
Train AUC for fold 1 : 0.962996
Train Mean AUC 0.960528 ± 0.000856
Val AUC for fold 0 : 0.859018
Val AUC for fold 1 : 0.857329
Val Mean AUC 0.857778 ± 0.000844
Random Guess
Train Around the mean TPRs
Val Around the mean TPRs

ROC - ROC Plot: Train & Validation of Ensemble (max_depth:20, num_estimators:20)
True Positive Rates
False Positive Rates
0.0
0.2
0.4
0.6
0.8
1.0
Train AUC for fold 0 : 0.999965
Train AUC for fold 1 : 0.999987
Train Mean AUC 0.994945 ± 0.000011
Val AUC for fold 0 : 0.826690
Val AUC for fold 1 : 0.804759
Val Mean AUC 0.815656 ± 0.010966
Random Guess
Train Around the mean TPRs
Val Around the mean TPRs

ROC - ROC Plot: Train & Validation of Ensemble (max_depth:20, num_estimators:60)
True Positive Rates
False Positive Rates
0.0
0.2
0.4
0.6
0.8
1.0
Train AUC for fold 0 : 1.000000
Train AUC for fold 1 : 0.999999
Train Mean AUC 0.994949 ± 0.000000
Val AUC for fold 0 : 0.805918
Val AUC for fold 1 : 0.818861
Val Mean AUC 0.812355 ± 0.006472
Random Guess
Train Around the mean TPRs
Val Around the mean TPRs

ROC - ROC Plot: Train & Validation of Ensemble (max_depth:20, num_estimators:120)
True Positive Rates
False Positive Rates
0.0
0.2
0.4
0.6
0.8
1.0
Train AUC for fold 0 : 1.000000
Train AUC for fold 1 : 1.000000
Train Mean AUC 0.994949 ± 0.000000
Val AUC for fold 0 : 0.812177
Val AUC for fold 1 : 0.805923
Val Mean AUC 0.808988 ± 0.003127
Random Guess
Train Around the mean TPRs
Val Around the mean TPRs

ROC - ROC Plot: Train & Validation of Ensemble (max_depth:20, num_estimators:300)
True Positive Rates
False Positive Rates
0.0
0.2
0.4
0.6
0.8
1.0
Train AUC for fold 0 : 1.000000
Train AUC for fold 1 : 1.000000
Train Mean AUC 0.994949 ± 0.000000
Val AUC for fold 0 : 0.798695
Val AUC for fold 1 : 0.782903
Val Mean AUC 0.790761 ± 0.007896
Random Guess
Train Around the mean TPRs
Val Around the mean TPRs

ROC - ROC Plot: Train & Validation of Ensemble (max_depth:20, num_estimators:500)
True Positive Rates
False Positive Rates
0.0
0.2
0.4
0.6
0.8
1.0
Train AUC for fold 0 : 1.000000
Train AUC for fold 1 : 1.000000
Train Mean AUC 0.994949 ± 0.000000
Val AUC for fold 0 : 0.786658
Val AUC for fold 1 : 0.799790
Val Mean AUC 0.793178 ± 0.006566
Random Guess
Train Around the mean TPRs
Val Around the mean TPRs

ROC - ROC Plot: Train & Validation of Ensemble (max_depth:50, num_estimators:20)
True Positive Rates
False Positive Rates
0.0
0.2
0.4
0.6
0.8
1.0
Train AUC for fold 0 : 0.999986
Train AUC for fold 1 : 0.999987
Train Mean AUC 0.994948 ± 0.000000
Val AUC for fold 0 : 0.823201
Val AUC for fold 1 : 0.825462
Val Mean AUC 0.824261 ± 0.001130
Random Guess
Train Around the mean TPRs
Val Around the mean TPRs

ROC - ROC Plot: Train & Validation of Ensemble (max_depth:50, num_estimators:60)
True Positive Rates
False Positive Rates
0.0
0.2
0.4
0.6
0.8
1.0
Train AUC for fold 0 : 1.000000
Train AUC for fold 1 : 1.000000
Train Mean AUC 0.994949 ± 0.000000
Val AUC for fold 0 : 0.776059
Val AUC for fold 1 : 0.801088
Val Mean AUC 0.788549 ± 0.012515
Random Guess
Train Around the mean TPRs
Val Around the mean TPRs

ROC - ROC Plot: Train & Validation of Ensemble (max_depth:50, num_estimators:120)
True Positive Rates
False Positive Rates
0.0
0.2
0.4
0.6
0.8
1.0
Train AUC for fold 0 : 1.000000
Train AUC for fold 1 : 1.000000
Train Mean AUC 0.994949 ± 0.000000
Val AUC for fold 0 : 0.795642
Val AUC for fold 1 : 0.820580
Val Mean AUC 0.808067 ± 0.012469
Random Guess
Train Around the mean TPRs
Val Around the mean TPRs

ROC - ROC Plot: Train & Validation of Ensemble (max_depth:50, num_estimators:300)
True Positive Rates
False Positive Rates
0.0
0.2
0.4
0.6
0.8
1.0
Train AUC for fold 0 : 1.000000
Train AUC for fold 1 : 1.000000
Train Mean AUC 0.994949 ± 0.000000
Val AUC for fold 0 : 0.842084
Val AUC for fold 1 : 0.841603
Val Mean AUC 0.841774 ± 0.000240
Random Guess
Train Around the mean TPRs
Val Around the mean TPRs

```
==========================================================================================
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++


 The depth vs AUC score plot
```

depth vs AUC
1.00
0.95
0.90
0.85
0.80
AUC Scores
Train AUC
Validation AUC
10
20
30
40
50
depth

```
++++++
```

num_estimators vs AUC
1.00
0.95
0.90
0.85
0.80
AUC Scores
Train AUC
Validation AUC
0
100
200
300
400
500
num_estimatores

++++++

Hyper param Heatmap DT-Train
max_depth
3
5
10
20
50
0.81 0.82 0.82 0.82 0.82
0.84 0.85 0.85 0.85 0.85
0.95 0.96 0.96 0.96 0.96
0.99 0.99 0.99 0.99 0.99
0.99 0.99 0.99 0.99 0.99
20 60 120 300 500
n_estimators
0.96
0.92
0.88
0.84

Hyper param Heatmap DT-Validation
max_depth
3
5
10
20
50
0.79 0.81 0.81 0.81 0.81
0.82 0.83 0.83 0.83 0.83
0.84 0.85 0.86 0.86 0.86
0.82 0.81 0.81 0.79 0.79
0.82 0.79 0.81 0.84 0.85
20 60 120 300 500
n_estimators
0.855
0.840
0.825
0.810
0.795

```
Best hyperparam value:  (10, 500)
```

	Negative	Positive
Precision	0.859409	0.866348
Recall	0.867993	0.857675
Fscore	0.863680	0.861990
Support	25029.000000	24971.000000

```
Test auc score  0.8611460084466451
```

```
              Negative      Positive
Precision     0.413173      0.947190
Recall        0.798563      0.761077
Fscore        0.544582      0.843995
Support    3480.000000  16520.000000
Results Summary:
 [('Hyper Param', (10, 500)), ('AUC', '0.8611'), ('f-score(-ve)', '54.4582'), ('f-score(+ve)', '
```

best hyper param identified is max_depth = 10, and n_estimators=500

4.5 [B] Applying GBDT using XGBOOST

```
In [22]: # In this work, Hyper parameter tuning is done only on the max_depth and n_estimators p
         # restricting to two are, adding more parameters will increase the training time expone
         # large dataset size. The other parameter values are left to its default values.
```

4.5.1 [B.1] Applying XGBOOST on BOW, SET 1

```
In [23]: # form two lists
         depth_list = [2, 5, 20, 50] # depends on size of dataset
```

```
n_estimators_list = [40, 70, 120, 200] # depends on size of dataset

# create a configuartion dictionary
config_dict = {
    'train_csv_path' : '/home/amd_3/AAIC/ASM_REPO/Processed_data/AMZN_FOOD_REVIW/BOW/tr
    'test_csv_path' : '/home/amd_3/AAIC/ASM_REPO/Processed_data/AMZN_FOOD_REVIW/BOW/tes
    'train_size' : 40000,
    'test_size' : 15000,
    'hyperparam_list' : list(product(depth_list, n_estimators_list)),
    'implementation': 'xgb' # 'xgb' or 'rf'
}
```

In [24]:
```
# read the train, test data and preprocess it
train_features, train_labels, test_features, test_labels = preprocess_data(config_dict,
                                                                           scaling=True
                                                                           dim_reductio

# train and validate the model
model = train_and_validate_model(config_dict, train_features, train_labels)

# test and evaluate the model
ptabe_entry_b1 = test_and_evaluate_model(config_dict, model, test_features, test_labels
```

```
Train df shape (40000, 503)
Class label distribution in train df:
 0    20024
1    19976
Name: Label, dtype: int64
Test df shape (15000, 503)
Class label distribution in test df:
 1    12386
0     2614
Name: Label, dtype: int64
Shape of ->  train features :40000,501, test features: 15000,501
Shape of -> train labels :40000, test labels: 15000
==========================================================================================


/home/amd_3/anaconda3/lib/python3.6/site-packages/sklearn/preprocessing/label.py:151: Deprecatio
  if diff:
/home/amd_3/anaconda3/lib/python3.6/site-packages/sklearn/preprocessing/label.py:151: Deprecatio
  if diff:
/home/amd_3/anaconda3/lib/python3.6/site-packages/sklearn/preprocessing/label.py:151: Deprecatio
  if diff:
/home/amd_3/anaconda3/lib/python3.6/site-packages/sklearn/preprocessing/label.py:151: Deprecatio
  if diff:
```

```
/home/amd_3/anaconda3/lib/python3.6/site-packages/sklearn/preprocessing/label.py:151: Deprecatio
  if diff:
/home/amd_3/anaconda3/lib/python3.6/site-packages/sklearn/preprocessing/label.py:151: Deprecatio
  if diff:
/home/amd_3/anaconda3/lib/python3.6/site-packages/sklearn/preprocessing/label.py:151: Deprecatio
  if diff:
/home/amd_3/anaconda3/lib/python3.6/site-packages/sklearn/preprocessing/label.py:151: Deprecatio
  if diff:
```

```
/home/amd_3/anaconda3/lib/python3.6/site-packages/sklearn/preprocessing/label.py:151: Deprecatio
  if diff:
/home/amd_3/anaconda3/lib/python3.6/site-packages/sklearn/preprocessing/label.py:151: Deprecatio
  if diff:
/home/amd_3/anaconda3/lib/python3.6/site-packages/sklearn/preprocessing/label.py:151: Deprecatio
  if diff:
/home/amd_3/anaconda3/lib/python3.6/site-packages/sklearn/preprocessing/label.py:151: Deprecatio
  if diff:
```

```
/home/amd_3/anaconda3/lib/python3.6/site-packages/sklearn/preprocessing/label.py:151: Deprecatio
  if diff:
/home/amd_3/anaconda3/lib/python3.6/site-packages/sklearn/preprocessing/label.py:151: Deprecatio
  if diff:
/home/amd_3/anaconda3/lib/python3.6/site-packages/sklearn/preprocessing/label.py:151: Deprecatio
  if diff:
/home/amd_3/anaconda3/lib/python3.6/site-packages/sklearn/preprocessing/label.py:151: Deprecatio
  if diff:
```

```
/home/amd_3/anaconda3/lib/python3.6/site-packages/sklearn/preprocessing/label.py:151: Deprecatio
  if diff:
/home/amd_3/anaconda3/lib/python3.6/site-packages/sklearn/preprocessing/label.py:151: Deprecatio
  if diff:
/home/amd_3/anaconda3/lib/python3.6/site-packages/sklearn/preprocessing/label.py:151: Deprecatio
  if diff:
/home/amd_3/anaconda3/lib/python3.6/site-packages/sklearn/preprocessing/label.py:151: Deprecatio
  if diff:
```

```
/home/amd_3/anaconda3/lib/python3.6/site-packages/sklearn/preprocessing/label.py:151: Deprecatio
  if diff:
/home/amd_3/anaconda3/lib/python3.6/site-packages/sklearn/preprocessing/label.py:151: Deprecatio
  if diff:
/home/amd_3/anaconda3/lib/python3.6/site-packages/sklearn/preprocessing/label.py:151: Deprecatio
  if diff:
/home/amd_3/anaconda3/lib/python3.6/site-packages/sklearn/preprocessing/label.py:151: Deprecatio
  if diff:
```

```
/home/amd_3/anaconda3/lib/python3.6/site-packages/sklearn/preprocessing/label.py:151: Deprecatio
  if diff:
/home/amd_3/anaconda3/lib/python3.6/site-packages/sklearn/preprocessing/label.py:151: Deprecatio
  if diff:
/home/amd_3/anaconda3/lib/python3.6/site-packages/sklearn/preprocessing/label.py:151: Deprecatio
  if diff:
/home/amd_3/anaconda3/lib/python3.6/site-packages/sklearn/preprocessing/label.py:151: Deprecatio
  if diff:
```

```
/home/amd_3/anaconda3/lib/python3.6/site-packages/sklearn/preprocessing/label.py:151: Deprecatio
  if diff:
/home/amd_3/anaconda3/lib/python3.6/site-packages/sklearn/preprocessing/label.py:151: Deprecatio
  if diff:
/home/amd_3/anaconda3/lib/python3.6/site-packages/sklearn/preprocessing/label.py:151: Deprecatio
  if diff:
/home/amd_3/anaconda3/lib/python3.6/site-packages/sklearn/preprocessing/label.py:151: Deprecatio
  if diff:
```

```
/home/amd_3/anaconda3/lib/python3.6/site-packages/sklearn/preprocessing/label.py:151: Deprecatio
  if diff:
/home/amd_3/anaconda3/lib/python3.6/site-packages/sklearn/preprocessing/label.py:151: Deprecatio
  if diff:
/home/amd_3/anaconda3/lib/python3.6/site-packages/sklearn/preprocessing/label.py:151: Deprecatio
  if diff:
/home/amd_3/anaconda3/lib/python3.6/site-packages/sklearn/preprocessing/label.py:151: Deprecatio
  if diff:
```

```
/home/amd_3/anaconda3/lib/python3.6/site-packages/sklearn/preprocessing/label.py:151: Deprecatio
  if diff:
/home/amd_3/anaconda3/lib/python3.6/site-packages/sklearn/preprocessing/label.py:151: Deprecatio
  if diff:
/home/amd_3/anaconda3/lib/python3.6/site-packages/sklearn/preprocessing/label.py:151: Deprecatio
  if diff:
/home/amd_3/anaconda3/lib/python3.6/site-packages/sklearn/preprocessing/label.py:151: Deprecatio
  if diff:
```

```
/home/amd_3/anaconda3/lib/python3.6/site-packages/sklearn/preprocessing/label.py:151: Deprecatio
  if diff:
/home/amd_3/anaconda3/lib/python3.6/site-packages/sklearn/preprocessing/label.py:151: Deprecatio
  if diff:
/home/amd_3/anaconda3/lib/python3.6/site-packages/sklearn/preprocessing/label.py:151: Deprecatio
  if diff:
/home/amd_3/anaconda3/lib/python3.6/site-packages/sklearn/preprocessing/label.py:151: Deprecatio
  if diff:
```

```
/home/amd_3/anaconda3/lib/python3.6/site-packages/sklearn/preprocessing/label.py:151: Deprecatio
  if diff:
/home/amd_3/anaconda3/lib/python3.6/site-packages/sklearn/preprocessing/label.py:151: Deprecatio
  if diff:
/home/amd_3/anaconda3/lib/python3.6/site-packages/sklearn/preprocessing/label.py:151: Deprecatio
  if diff:
/home/amd_3/anaconda3/lib/python3.6/site-packages/sklearn/preprocessing/label.py:151: Deprecatio
  if diff:
```

```
/home/amd_3/anaconda3/lib/python3.6/site-packages/sklearn/preprocessing/label.py:151: Deprecatio
  if diff:
/home/amd_3/anaconda3/lib/python3.6/site-packages/sklearn/preprocessing/label.py:151: Deprecatio
  if diff:
/home/amd_3/anaconda3/lib/python3.6/site-packages/sklearn/preprocessing/label.py:151: Deprecatio
  if diff:
/home/amd_3/anaconda3/lib/python3.6/site-packages/sklearn/preprocessing/label.py:151: Deprecatio
  if diff:
```

```
/home/amd_3/anaconda3/lib/python3.6/site-packages/sklearn/preprocessing/label.py:151: Deprecatio
  if diff:
/home/amd_3/anaconda3/lib/python3.6/site-packages/sklearn/preprocessing/label.py:151: Deprecatio
  if diff:
/home/amd_3/anaconda3/lib/python3.6/site-packages/sklearn/preprocessing/label.py:151: Deprecatio
  if diff:
/home/amd_3/anaconda3/lib/python3.6/site-packages/sklearn/preprocessing/label.py:151: Deprecatio
  if diff:
```

```
/home/amd_3/anaconda3/lib/python3.6/site-packages/sklearn/preprocessing/label.py:151: Deprecatio
  if diff:
/home/amd_3/anaconda3/lib/python3.6/site-packages/sklearn/preprocessing/label.py:151: Deprecatio
  if diff:
/home/amd_3/anaconda3/lib/python3.6/site-packages/sklearn/preprocessing/label.py:151: Deprecatio
  if diff:
/home/amd_3/anaconda3/lib/python3.6/site-packages/sklearn/preprocessing/label.py:151: Deprecatio
  if diff:
```

```
/home/amd_3/anaconda3/lib/python3.6/site-packages/sklearn/preprocessing/label.py:151: Deprecatio
  if diff:
/home/amd_3/anaconda3/lib/python3.6/site-packages/sklearn/preprocessing/label.py:151: Deprecatio
  if diff:
/home/amd_3/anaconda3/lib/python3.6/site-packages/sklearn/preprocessing/label.py:151: Deprecatio
  if diff:
/home/amd_3/anaconda3/lib/python3.6/site-packages/sklearn/preprocessing/label.py:151: Deprecatio
  if diff:
```

```
==========================================================================================
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++


 The depth vs AUC score plot
```

depth vs AUC
1.00
0.95
0.90
0.85
AUC Scores
0
10
20
30
40
50
depth
Train AUC
Validation AUC

```
++++++
```

num_estimators vs AUC
1.00
0.95
0.90
0.85
AUC Scores
50
100
150
200
num_estimatores
Train AUC
Validation AUC

++++++

Hyper param Heatmap DT-Train
max_depth
2
5
20
50
0.83
0.86
0.88
0.9
0.89
0.92
0.94
0.96
0.99
0.99
0.99
0.99
0.99
0.99
0.99
0.99
40
70
120
200
n_estimators
0.99
0.96
0.93
0.90
0.87
0.84

Hyper param Heatmap DT-Validation
max_depth
2
5
20
50
0.83
0.85
0.88
0.89
0.87
0.89
0.91
0.92
0.9
0.91
0.92
0.92
0.9
0.91
0.92
0.92
40
70
120
200
n_estimators
0.92
0.90
0.88
0.86
0.84

```
Best hyperparam value:  (20, 200)
```

```
/home/amd_3/anaconda3/lib/python3.6/site-packages/sklearn/preprocessing/label.py:151: Deprecatio
  if diff:
```

	Negative	Positive
Precision	0.985086	0.989539
Recall	0.989612	0.984982
Fscore	0.987344	0.987255
Support	20024.000000	19976.000000

```
/home/amd_3/anaconda3/lib/python3.6/site-packages/sklearn/preprocessing/label.py:151: Deprecatio
  if diff:
```

```
Test auc score  0.932363355825329
```

```
              Negative      Positive
Precision     0.525229      0.969549
Recall        0.876052      0.832876
Fscore        0.656725      0.896031
Support    2614.000000  12386.000000
Results Summary:
 [('Hyper Param', (20, 200)), ('AUC', '0.9324'), ('f-score(-ve)', '65.6725'), ('f-score(+ve)', '
```

4.5.2 [B.2] Applying XGBOOST on TFIDF, SET 2

```
In [25]: # form two lists
         depth_list = [2, 5, 20, 50] # depends on size of dataset
         n_estimators_listt = [40, 70, 120, 200] # depends on size of dataset

         # create a configuartion dictionary
         config_dict = {
             'train_csv_path' : '/home/amd_3/AAIC/ASM_REPO/Processed_data/AMZN_FOOD_REVIW/TFIDF/
             'test_csv_path' : '/home/amd_3/AAIC/ASM_REPO/Processed_data/AMZN_FOOD_REVIW/TFIDF/t
             'train_size' : 40000,
```

```
            'test_size' : 15000,
            'hyperparam_list' : list(product(depth_list, n_estimators_list)),
            'implementation': 'xgb' # 'xgb' or 'rf'
        }
```

In [26]:
```
# read the train, test data and preprocess it
train_features, train_labels, test_features, test_labels = preprocess_data(config_dict,
                                                                           scaling=True
                                                                           dim_reductio

# train and validate the model
model = train_and_validate_model(config_dict, train_features, train_labels)

# test and evaluate the model
ptabe_entry_b2 = test_and_evaluate_model(config_dict, model, test_features, test_labels
```

```
Train df shape (40000, 503)
Class label distribution in train df:
 0    20024
1    19976
Name: Label, dtype: int64
Test df shape (15000, 503)
Class label distribution in test df:
 1    12386
0     2614
Name: Label, dtype: int64
Shape of ->  train features :40000,501, test features: 15000,501
Shape of -> train labels :40000, test labels: 15000
==========================================================================================


/home/amd_3/anaconda3/lib/python3.6/site-packages/sklearn/preprocessing/label.py:151: Deprecatio
  if diff:
/home/amd_3/anaconda3/lib/python3.6/site-packages/sklearn/preprocessing/label.py:151: Deprecatio
  if diff:
/home/amd_3/anaconda3/lib/python3.6/site-packages/sklearn/preprocessing/label.py:151: Deprecatio
  if diff:
/home/amd_3/anaconda3/lib/python3.6/site-packages/sklearn/preprocessing/label.py:151: Deprecatio
  if diff:
```

```
/home/amd_3/anaconda3/lib/python3.6/site-packages/sklearn/preprocessing/label.py:151: Deprecatio
  if diff:
/home/amd_3/anaconda3/lib/python3.6/site-packages/sklearn/preprocessing/label.py:151: Deprecatio
  if diff:
/home/amd_3/anaconda3/lib/python3.6/site-packages/sklearn/preprocessing/label.py:151: Deprecatio
  if diff:
/home/amd_3/anaconda3/lib/python3.6/site-packages/sklearn/preprocessing/label.py:151: Deprecatio
  if diff:
```

```
/home/amd_3/anaconda3/lib/python3.6/site-packages/sklearn/preprocessing/label.py:151: Deprecatio
  if diff:
/home/amd_3/anaconda3/lib/python3.6/site-packages/sklearn/preprocessing/label.py:151: Deprecatio
  if diff:
/home/amd_3/anaconda3/lib/python3.6/site-packages/sklearn/preprocessing/label.py:151: Deprecatio
  if diff:
/home/amd_3/anaconda3/lib/python3.6/site-packages/sklearn/preprocessing/label.py:151: Deprecatio
  if diff:
```

```
/home/amd_3/anaconda3/lib/python3.6/site-packages/sklearn/preprocessing/label.py:151: Deprecatio
  if diff:
/home/amd_3/anaconda3/lib/python3.6/site-packages/sklearn/preprocessing/label.py:151: Deprecatio
  if diff:
/home/amd_3/anaconda3/lib/python3.6/site-packages/sklearn/preprocessing/label.py:151: Deprecatio
  if diff:
/home/amd_3/anaconda3/lib/python3.6/site-packages/sklearn/preprocessing/label.py:151: Deprecatio
  if diff:
```

```
/home/amd_3/anaconda3/lib/python3.6/site-packages/sklearn/preprocessing/label.py:151: Deprecatio
  if diff:
/home/amd_3/anaconda3/lib/python3.6/site-packages/sklearn/preprocessing/label.py:151: Deprecatio
  if diff:
/home/amd_3/anaconda3/lib/python3.6/site-packages/sklearn/preprocessing/label.py:151: Deprecatio
  if diff:
/home/amd_3/anaconda3/lib/python3.6/site-packages/sklearn/preprocessing/label.py:151: Deprecatio
  if diff:
```

```
/home/amd_3/anaconda3/lib/python3.6/site-packages/sklearn/preprocessing/label.py:151: Deprecatio
  if diff:
/home/amd_3/anaconda3/lib/python3.6/site-packages/sklearn/preprocessing/label.py:151: Deprecatio
  if diff:
/home/amd_3/anaconda3/lib/python3.6/site-packages/sklearn/preprocessing/label.py:151: Deprecatio
  if diff:
/home/amd_3/anaconda3/lib/python3.6/site-packages/sklearn/preprocessing/label.py:151: Deprecatio
  if diff:
```

```
/home/amd_3/anaconda3/lib/python3.6/site-packages/sklearn/preprocessing/label.py:151: Deprecatio
  if diff:
/home/amd_3/anaconda3/lib/python3.6/site-packages/sklearn/preprocessing/label.py:151: Deprecatio
  if diff:
/home/amd_3/anaconda3/lib/python3.6/site-packages/sklearn/preprocessing/label.py:151: Deprecatio
  if diff:
/home/amd_3/anaconda3/lib/python3.6/site-packages/sklearn/preprocessing/label.py:151: Deprecatio
  if diff:
```

```
/home/amd_3/anaconda3/lib/python3.6/site-packages/sklearn/preprocessing/label.py:151: Deprecatio
  if diff:
/home/amd_3/anaconda3/lib/python3.6/site-packages/sklearn/preprocessing/label.py:151: Deprecatio
  if diff:
/home/amd_3/anaconda3/lib/python3.6/site-packages/sklearn/preprocessing/label.py:151: Deprecatio
  if diff:
/home/amd_3/anaconda3/lib/python3.6/site-packages/sklearn/preprocessing/label.py:151: Deprecatio
  if diff:
```

```
/home/amd_3/anaconda3/lib/python3.6/site-packages/sklearn/preprocessing/label.py:151: Deprecatio
  if diff:
/home/amd_3/anaconda3/lib/python3.6/site-packages/sklearn/preprocessing/label.py:151: Deprecatio
  if diff:
/home/amd_3/anaconda3/lib/python3.6/site-packages/sklearn/preprocessing/label.py:151: Deprecatio
  if diff:
/home/amd_3/anaconda3/lib/python3.6/site-packages/sklearn/preprocessing/label.py:151: Deprecatio
  if diff:
```

```
/home/amd_3/anaconda3/lib/python3.6/site-packages/sklearn/preprocessing/label.py:151: Deprecatio
  if diff:
/home/amd_3/anaconda3/lib/python3.6/site-packages/sklearn/preprocessing/label.py:151: Deprecatio
  if diff:
/home/amd_3/anaconda3/lib/python3.6/site-packages/sklearn/preprocessing/label.py:151: Deprecatio
  if diff:
/home/amd_3/anaconda3/lib/python3.6/site-packages/sklearn/preprocessing/label.py:151: Deprecatio
  if diff:
```

```
/home/amd_3/anaconda3/lib/python3.6/site-packages/sklearn/preprocessing/label.py:151: Deprecatio
  if diff:
/home/amd_3/anaconda3/lib/python3.6/site-packages/sklearn/preprocessing/label.py:151: Deprecatio
  if diff:
/home/amd_3/anaconda3/lib/python3.6/site-packages/sklearn/preprocessing/label.py:151: Deprecatio
  if diff:
/home/amd_3/anaconda3/lib/python3.6/site-packages/sklearn/preprocessing/label.py:151: Deprecatio
  if diff:
```

```
/home/amd_3/anaconda3/lib/python3.6/site-packages/sklearn/preprocessing/label.py:151: Deprecatio
  if diff:
/home/amd_3/anaconda3/lib/python3.6/site-packages/sklearn/preprocessing/label.py:151: Deprecatio
  if diff:
/home/amd_3/anaconda3/lib/python3.6/site-packages/sklearn/preprocessing/label.py:151: Deprecatio
  if diff:
/home/amd_3/anaconda3/lib/python3.6/site-packages/sklearn/preprocessing/label.py:151: Deprecatio
  if diff:
```

```
/home/amd_3/anaconda3/lib/python3.6/site-packages/sklearn/preprocessing/label.py:151: Deprecatio
  if diff:
/home/amd_3/anaconda3/lib/python3.6/site-packages/sklearn/preprocessing/label.py:151: Deprecatio
  if diff:
/home/amd_3/anaconda3/lib/python3.6/site-packages/sklearn/preprocessing/label.py:151: Deprecatio
  if diff:
/home/amd_3/anaconda3/lib/python3.6/site-packages/sklearn/preprocessing/label.py:151: Deprecatio
  if diff:
```

```
/home/amd_3/anaconda3/lib/python3.6/site-packages/sklearn/preprocessing/label.py:151: Deprecatio
  if diff:
/home/amd_3/anaconda3/lib/python3.6/site-packages/sklearn/preprocessing/label.py:151: Deprecatio
  if diff:
/home/amd_3/anaconda3/lib/python3.6/site-packages/sklearn/preprocessing/label.py:151: Deprecatio
  if diff:
/home/amd_3/anaconda3/lib/python3.6/site-packages/sklearn/preprocessing/label.py:151: Deprecatio
  if diff:
```

```
/home/amd_3/anaconda3/lib/python3.6/site-packages/sklearn/preprocessing/label.py:151: Deprecatio
  if diff:
/home/amd_3/anaconda3/lib/python3.6/site-packages/sklearn/preprocessing/label.py:151: Deprecatio
  if diff:
/home/amd_3/anaconda3/lib/python3.6/site-packages/sklearn/preprocessing/label.py:151: Deprecatio
  if diff:
/home/amd_3/anaconda3/lib/python3.6/site-packages/sklearn/preprocessing/label.py:151: Deprecatio
  if diff:
```

```
/home/amd_3/anaconda3/lib/python3.6/site-packages/sklearn/preprocessing/label.py:151: Deprecatio
  if diff:
/home/amd_3/anaconda3/lib/python3.6/site-packages/sklearn/preprocessing/label.py:151: Deprecatio
  if diff:
/home/amd_3/anaconda3/lib/python3.6/site-packages/sklearn/preprocessing/label.py:151: Deprecatio
  if diff:
/home/amd_3/anaconda3/lib/python3.6/site-packages/sklearn/preprocessing/label.py:151: Deprecatio
  if diff:
```

ROC - ROC Plot: Train & Validation of Ensemble (max_depth:50, num_estimators:200)
True Positive Rates
False Positive Rates
0.0
0.2
0.4
0.6
0.8
1.0
Train AUC for fold 0 : 1.000000
Train AUC for fold 1 : 0.999999
Train Mean AUC 0.994949 ± 0.000000
Val AUC for fold 0 : 0.920282
Val AUC for fold 1 : 0.921868
Val Mean AUC 0.920279 ± 0.000793
Random Guess
Train Around the mean TPRs
Val Around the mean TPRs

==========
++++++

depth vs AUC
1.00
0.95
0.90
0.85
AUC Scores
0
10
20
30
40
50
depth
Train AUC
Validation AUC

```
++++++
```

num_estimators vs AUC
1.00
0.95
0.90
0.85
AUC Scores
50
100
150
200
num_estimatores
Train AUC
Validation AUC

++++++

Hyper param Heatmap DT-Train
max_depth
2
5
20
50
0.84
0.86
0.89
0.91
0.9
0.93
0.95
0.96
0.99
0.99
0.99
0.99
0.99
0.99
0.99
0.99
40
70
120
200
n_estimators
0.99
0.96
0.93
0.90
0.87
0.84

Hyper param Heatmap DT-Validation
max_depth
2
5
20
50
0.83
0.86
0.88
0.9
0.87
0.89
0.91
0.92
0.9
0.91
0.92
0.92
0.9
0.91
0.92
0.92
40
70
120
200
n_estimators
0.92
0.90
0.88
0.86
0.84

```
Best hyperparam value:  (20, 200)
```

```
/home/amd_3/anaconda3/lib/python3.6/site-packages/sklearn/preprocessing/label.py:151: Deprecatio
  if diff:
```

	Negative	Positive
Precision	0.989815	0.994867
Recall	0.994906	0.989738
Fscore	0.992354	0.992296
Support	20024.000000	19976.000000

```
/home/amd_3/anaconda3/lib/python3.6/site-packages/sklearn/preprocessing/label.py:151: Deprecatio
  if diff:
```

```
Test auc score  0.9305170223142127
```

```
              Negative      Positive
Precision     0.521719      0.968756
Recall        0.872992      0.831100
Fscore        0.653120      0.894664
Support    2614.000000  12386.000000
Results Summary:
 [('Hyper Param', (20, 200)), ('AUC', '0.9305'), ('f-score(-ve)', '65.3120'), ('f-score(+ve)', '
```

4.5.3 [B.3] Applying XGBOOST on AVG W2V, SET 3

```
In [27]: # form two lists
         depth_list = [2, 5, 20, 50] # depends on size of dataset
         n_estimators_list = [40, 70, 120, 200] # depends on size of dataset

         # create a configuartion dictionary
         config_dict = {
             'train_csv_path' : '/home/amd_3/AAIC/ASM_REPO/Processed_data/AMZN_FOOD_REVIW/AVG_W2
             'test_csv_path' : '/home/amd_3/AAIC/ASM_REPO/Processed_data/AMZN_FOOD_REVIW/AVG_W2V
             'train_size' : 40000,
```

```
            'test_size' : 15000,
            'hyperparam_list' : list(product(depth_list, n_estimators_list)),
            'implementation': 'xgb' # 'xgb' or 'rf'
        }
```

In [28]:
```
# read the train, test data and preprocess it
train_features, train_labels, test_features, test_labels = preprocess_data(config_dict,
                                                                           scaling=True
                                                                           dim_reductio

# train and validate the model
model = train_and_validate_model(config_dict, train_features, train_labels)

# test and evaluate the model
ptabe_entry_b3 = test_and_evaluate_model(config_dict, model, test_features, test_labels
```

```
Train df shape (40000, 52)
Class label distribution in train df:
 0    20024
1    19976
Name: Label, dtype: int64
Test df shape (15000, 52)
Class label distribution in test df:
 1    12386
0     2614
Name: Label, dtype: int64
Shape of ->  train features :40000,50, test features: 15000,50
Shape of -> train labels :40000, test labels: 15000
==========================================================================================


/home/amd_3/anaconda3/lib/python3.6/site-packages/sklearn/preprocessing/label.py:151: Deprecatio
  if diff:
/home/amd_3/anaconda3/lib/python3.6/site-packages/sklearn/preprocessing/label.py:151: Deprecatio
  if diff:
/home/amd_3/anaconda3/lib/python3.6/site-packages/sklearn/preprocessing/label.py:151: Deprecatio
  if diff:
/home/amd_3/anaconda3/lib/python3.6/site-packages/sklearn/preprocessing/label.py:151: Deprecatio
  if diff:
```

```
/home/amd_3/anaconda3/lib/python3.6/site-packages/sklearn/preprocessing/label.py:151: Deprecatio
  if diff:
/home/amd_3/anaconda3/lib/python3.6/site-packages/sklearn/preprocessing/label.py:151: Deprecatio
  if diff:
/home/amd_3/anaconda3/lib/python3.6/site-packages/sklearn/preprocessing/label.py:151: Deprecatio
  if diff:
/home/amd_3/anaconda3/lib/python3.6/site-packages/sklearn/preprocessing/label.py:151: Deprecatio
  if diff:
```

```
/home/amd_3/anaconda3/lib/python3.6/site-packages/sklearn/preprocessing/label.py:151: Deprecatio
  if diff:
/home/amd_3/anaconda3/lib/python3.6/site-packages/sklearn/preprocessing/label.py:151: Deprecatio
  if diff:
/home/amd_3/anaconda3/lib/python3.6/site-packages/sklearn/preprocessing/label.py:151: Deprecatio
  if diff:
/home/amd_3/anaconda3/lib/python3.6/site-packages/sklearn/preprocessing/label.py:151: Deprecatio
  if diff:
```

```
/home/amd_3/anaconda3/lib/python3.6/site-packages/sklearn/preprocessing/label.py:151: Deprecatio
  if diff:
/home/amd_3/anaconda3/lib/python3.6/site-packages/sklearn/preprocessing/label.py:151: Deprecatio
  if diff:
/home/amd_3/anaconda3/lib/python3.6/site-packages/sklearn/preprocessing/label.py:151: Deprecatio
  if diff:
/home/amd_3/anaconda3/lib/python3.6/site-packages/sklearn/preprocessing/label.py:151: Deprecatio
  if diff:
```

```
/home/amd_3/anaconda3/lib/python3.6/site-packages/sklearn/preprocessing/label.py:151: Deprecatio
  if diff:
/home/amd_3/anaconda3/lib/python3.6/site-packages/sklearn/preprocessing/label.py:151: Deprecatio
  if diff:
/home/amd_3/anaconda3/lib/python3.6/site-packages/sklearn/preprocessing/label.py:151: Deprecatio
  if diff:
/home/amd_3/anaconda3/lib/python3.6/site-packages/sklearn/preprocessing/label.py:151: Deprecatio
  if diff:
```

```
/home/amd_3/anaconda3/lib/python3.6/site-packages/sklearn/preprocessing/label.py:151: Deprecatio
  if diff:
/home/amd_3/anaconda3/lib/python3.6/site-packages/sklearn/preprocessing/label.py:151: Deprecatio
  if diff:
/home/amd_3/anaconda3/lib/python3.6/site-packages/sklearn/preprocessing/label.py:151: Deprecatio
  if diff:
/home/amd_3/anaconda3/lib/python3.6/site-packages/sklearn/preprocessing/label.py:151: Deprecatio
  if diff:
```

```
/home/amd_3/anaconda3/lib/python3.6/site-packages/sklearn/preprocessing/label.py:151: Deprecatio
  if diff:
/home/amd_3/anaconda3/lib/python3.6/site-packages/sklearn/preprocessing/label.py:151: Deprecatio
  if diff:
/home/amd_3/anaconda3/lib/python3.6/site-packages/sklearn/preprocessing/label.py:151: Deprecatio
  if diff:
/home/amd_3/anaconda3/lib/python3.6/site-packages/sklearn/preprocessing/label.py:151: Deprecatio
  if diff:
```

```
/home/amd_3/anaconda3/lib/python3.6/site-packages/sklearn/preprocessing/label.py:151: Deprecatio
  if diff:
/home/amd_3/anaconda3/lib/python3.6/site-packages/sklearn/preprocessing/label.py:151: Deprecatio
  if diff:
/home/amd_3/anaconda3/lib/python3.6/site-packages/sklearn/preprocessing/label.py:151: Deprecatio
  if diff:
/home/amd_3/anaconda3/lib/python3.6/site-packages/sklearn/preprocessing/label.py:151: Deprecatio
  if diff:
```

```
/home/amd_3/anaconda3/lib/python3.6/site-packages/sklearn/preprocessing/label.py:151: Deprecatio
  if diff:
/home/amd_3/anaconda3/lib/python3.6/site-packages/sklearn/preprocessing/label.py:151: Deprecatio
  if diff:
/home/amd_3/anaconda3/lib/python3.6/site-packages/sklearn/preprocessing/label.py:151: Deprecatio
  if diff:
/home/amd_3/anaconda3/lib/python3.6/site-packages/sklearn/preprocessing/label.py:151: Deprecatio
  if diff:
```

```
/home/amd_3/anaconda3/lib/python3.6/site-packages/sklearn/preprocessing/label.py:151: Deprecatio
  if diff:
/home/amd_3/anaconda3/lib/python3.6/site-packages/sklearn/preprocessing/label.py:151: Deprecatio
  if diff:
/home/amd_3/anaconda3/lib/python3.6/site-packages/sklearn/preprocessing/label.py:151: Deprecatio
  if diff:
/home/amd_3/anaconda3/lib/python3.6/site-packages/sklearn/preprocessing/label.py:151: Deprecatio
  if diff:
```

```
/home/amd_3/anaconda3/lib/python3.6/site-packages/sklearn/preprocessing/label.py:151: Deprecatio
  if diff:
/home/amd_3/anaconda3/lib/python3.6/site-packages/sklearn/preprocessing/label.py:151: Deprecatio
  if diff:
/home/amd_3/anaconda3/lib/python3.6/site-packages/sklearn/preprocessing/label.py:151: Deprecatio
  if diff:
/home/amd_3/anaconda3/lib/python3.6/site-packages/sklearn/preprocessing/label.py:151: Deprecatio
  if diff:
```

```
/home/amd_3/anaconda3/lib/python3.6/site-packages/sklearn/preprocessing/label.py:151: Deprecatio
  if diff:
/home/amd_3/anaconda3/lib/python3.6/site-packages/sklearn/preprocessing/label.py:151: Deprecatio
  if diff:
/home/amd_3/anaconda3/lib/python3.6/site-packages/sklearn/preprocessing/label.py:151: Deprecatio
  if diff:
/home/amd_3/anaconda3/lib/python3.6/site-packages/sklearn/preprocessing/label.py:151: Deprecatio
  if diff:
```

```
/home/amd_3/anaconda3/lib/python3.6/site-packages/sklearn/preprocessing/label.py:151: Deprecatio
  if diff:
/home/amd_3/anaconda3/lib/python3.6/site-packages/sklearn/preprocessing/label.py:151: Deprecatio
  if diff:
/home/amd_3/anaconda3/lib/python3.6/site-packages/sklearn/preprocessing/label.py:151: Deprecatio
  if diff:
/home/amd_3/anaconda3/lib/python3.6/site-packages/sklearn/preprocessing/label.py:151: Deprecatio
  if diff:
```

```
/home/amd_3/anaconda3/lib/python3.6/site-packages/sklearn/preprocessing/label.py:151: Deprecatio
  if diff:
/home/amd_3/anaconda3/lib/python3.6/site-packages/sklearn/preprocessing/label.py:151: Deprecatio
  if diff:
/home/amd_3/anaconda3/lib/python3.6/site-packages/sklearn/preprocessing/label.py:151: Deprecatio
  if diff:
/home/amd_3/anaconda3/lib/python3.6/site-packages/sklearn/preprocessing/label.py:151: Deprecatio
  if diff:
```

```
/home/amd_3/anaconda3/lib/python3.6/site-packages/sklearn/preprocessing/label.py:151: Deprecatio
  if diff:
/home/amd_3/anaconda3/lib/python3.6/site-packages/sklearn/preprocessing/label.py:151: Deprecatio
  if diff:
/home/amd_3/anaconda3/lib/python3.6/site-packages/sklearn/preprocessing/label.py:151: Deprecatio
  if diff:
/home/amd_3/anaconda3/lib/python3.6/site-packages/sklearn/preprocessing/label.py:151: Deprecatio
  if diff:
```

```
/home/amd_3/anaconda3/lib/python3.6/site-packages/sklearn/preprocessing/label.py:151: Deprecatio
  if diff:
/home/amd_3/anaconda3/lib/python3.6/site-packages/sklearn/preprocessing/label.py:151: Deprecatio
  if diff:
/home/amd_3/anaconda3/lib/python3.6/site-packages/sklearn/preprocessing/label.py:151: Deprecatio
  if diff:
/home/amd_3/anaconda3/lib/python3.6/site-packages/sklearn/preprocessing/label.py:151: Deprecatio
  if diff:
```

ROC - ROC Plot: Train & Validation of Ensemble (max_depth:50, num_estimators:200)
True Positive Rates
False Positive Rates
0.0
0.2
0.4
0.6
0.8
1.0
Train AUC for fold 0 : 1.000000
Train AUC for fold 1 : 1.000000
Train Mean AUC 0.994949 ± 0.000000
Val AUC for fold 0 : 0.921992
Val AUC for fold 1 : 0.919858
Val Mean AUC 0.919988 ± 0.001067
Random Guess
Train Around the mean TPRs
Val Around the mean TPRs

==========
++++++

depth vs AUC
1.00
0.98
0.96
0.94
0.92
0.90
0.88
AUC Scores
0
10
20
30
40
50
depth
Train AUC
Validation AUC

```
++++++
```

num_estimators vs AUC
1.00
0.98
0.96
0.94
0.92
0.90
0.88
AUC Scores
50
100
150
200
num_estimatores
Train AUC
Validation AUC

++++++

Hyper param Heatmap DT-Train
max_depth
2
5
20
50
0.89
0.9
0.92
0.93
0.94
0.95
0.97
0.98
0.99
0.99
0.99
0.99
0.99
0.99
0.99
0.99
40
70
120
200
n_estimators
0.98
0.96
0.94
0.92
0.90

Hyper param Heatmap DT-Validation
max_depth
2
5
20
50
0.88
0.89
0.9
0.91
0.91
0.91
0.92
0.92
0.91
0.91
0.92
0.92
0.91
0.91
0.92
0.92
40
70
120
200
n_estimators
0.920
0.912
0.904
0.896
0.888
0.880

```
Best hyperparam value:  (5, 200)
```

```
/home/amd_3/anaconda3/lib/python3.6/site-packages/sklearn/preprocessing/label.py:151: Deprecatio
  if diff:
```

	Negative	Positive
Precision	0.896322	0.904167
Recall	0.905364	0.895024
Fscore	0.900820	0.899572
Support	20024.000000	19976.000000

```
/home/amd_3/anaconda3/lib/python3.6/site-packages/sklearn/preprocessing/label.py:151: Deprecatio
  if diff:
```

```
Test auc score  0.9250815926094467
```

```
              Negative      Positive
Precision     0.512698      0.966667
Recall        0.864958      0.826498
Fscore        0.643793      0.891104
Support    2614.000000  12386.000000
Results Summary:
 [('Hyper Param', (5, 200)), ('AUC', '0.9251'), ('f-score(-ve)', '64.3793'), ('f-score(+ve)', '8
```

4.5.4 [B.4] Applying XGBOOST on TFIDF W2V, SET 4

```
In [29]: # form two lists
         depth_list = [2, 5, 20, 50] # depends on size of dataset
         n_estimators_list = [40, 70, 120, 200] # depends on size of dataset

         # create a configuarion dictionary
         config_dict = {
             'train_csv_path' : '/home/amd_3/AAIC/ASM_REPO/Processed_data/AMZN_FOOD_REVIW/TFIDF_
             'test_csv_path' : '/home/amd_3/AAIC/ASM_REPO/Processed_data/AMZN_FOOD_REVIW/TFIDF_W
             'train_size' : 40000,
```

```
            'test_size' : 15000,
            'hyperparam_list' : list(product(depth_list, n_estimators_list)),
            'implementation': 'xgb' # 'xgb' or 'rf'
        }
```

In [30]:
```
# read the train, test data and preprocess it
train_features, train_labels, test_features, test_labels = preprocess_data(config_dict,
                                                                           scaling=True
                                                                           dim_reductio

# train and validate the model
model = train_and_validate_model(config_dict, train_features, train_labels)

# test and evaluate the model
ptabe_entry_b4 = test_and_evaluate_model(config_dict, model, test_features, test_labels
```

```
Train df shape (40000, 52)
Class label distribution in train df:
 0    20024
1    19976
Name: Label, dtype: int64
Test df shape (15000, 52)
Class label distribution in test df:
 1    12386
0     2614
Name: Label, dtype: int64
Shape of ->  train features :40000,50, test features: 15000,50
Shape of -> train labels :40000, test labels: 15000
==========================================================================================


/home/amd_3/anaconda3/lib/python3.6/site-packages/sklearn/preprocessing/label.py:151: Deprecatio
  if diff:
/home/amd_3/anaconda3/lib/python3.6/site-packages/sklearn/preprocessing/label.py:151: Deprecatio
  if diff:
/home/amd_3/anaconda3/lib/python3.6/site-packages/sklearn/preprocessing/label.py:151: Deprecatio
  if diff:
/home/amd_3/anaconda3/lib/python3.6/site-packages/sklearn/preprocessing/label.py:151: Deprecatio
  if diff:
```

```
/home/amd_3/anaconda3/lib/python3.6/site-packages/sklearn/preprocessing/label.py:151: Deprecatio
  if diff:
/home/amd_3/anaconda3/lib/python3.6/site-packages/sklearn/preprocessing/label.py:151: Deprecatio
  if diff:
/home/amd_3/anaconda3/lib/python3.6/site-packages/sklearn/preprocessing/label.py:151: Deprecatio
  if diff:
/home/amd_3/anaconda3/lib/python3.6/site-packages/sklearn/preprocessing/label.py:151: Deprecatio
  if diff:
```

```
/home/amd_3/anaconda3/lib/python3.6/site-packages/sklearn/preprocessing/label.py:151: Deprecatio
  if diff:
/home/amd_3/anaconda3/lib/python3.6/site-packages/sklearn/preprocessing/label.py:151: Deprecatio
  if diff:
/home/amd_3/anaconda3/lib/python3.6/site-packages/sklearn/preprocessing/label.py:151: Deprecatio
  if diff:
/home/amd_3/anaconda3/lib/python3.6/site-packages/sklearn/preprocessing/label.py:151: Deprecatio
  if diff:
```

```
/home/amd_3/anaconda3/lib/python3.6/site-packages/sklearn/preprocessing/label.py:151: Deprecatio
  if diff:
/home/amd_3/anaconda3/lib/python3.6/site-packages/sklearn/preprocessing/label.py:151: Deprecatio
  if diff:
/home/amd_3/anaconda3/lib/python3.6/site-packages/sklearn/preprocessing/label.py:151: Deprecatio
  if diff:
/home/amd_3/anaconda3/lib/python3.6/site-packages/sklearn/preprocessing/label.py:151: Deprecatio
  if diff:
```

```
/home/amd_3/anaconda3/lib/python3.6/site-packages/sklearn/preprocessing/label.py:151: Deprecatio
  if diff:
/home/amd_3/anaconda3/lib/python3.6/site-packages/sklearn/preprocessing/label.py:151: Deprecatio
  if diff:
/home/amd_3/anaconda3/lib/python3.6/site-packages/sklearn/preprocessing/label.py:151: Deprecatio
  if diff:
/home/amd_3/anaconda3/lib/python3.6/site-packages/sklearn/preprocessing/label.py:151: Deprecatio
  if diff:
```

```
/home/amd_3/anaconda3/lib/python3.6/site-packages/sklearn/preprocessing/label.py:151: Deprecatio
  if diff:
/home/amd_3/anaconda3/lib/python3.6/site-packages/sklearn/preprocessing/label.py:151: Deprecatio
  if diff:
/home/amd_3/anaconda3/lib/python3.6/site-packages/sklearn/preprocessing/label.py:151: Deprecatio
  if diff:
/home/amd_3/anaconda3/lib/python3.6/site-packages/sklearn/preprocessing/label.py:151: Deprecatio
  if diff:
```

```
/home/amd_3/anaconda3/lib/python3.6/site-packages/sklearn/preprocessing/label.py:151: Deprecatio
  if diff:
/home/amd_3/anaconda3/lib/python3.6/site-packages/sklearn/preprocessing/label.py:151: Deprecatio
  if diff:
/home/amd_3/anaconda3/lib/python3.6/site-packages/sklearn/preprocessing/label.py:151: Deprecatio
  if diff:
/home/amd_3/anaconda3/lib/python3.6/site-packages/sklearn/preprocessing/label.py:151: Deprecatio
  if diff:
```

```
/home/amd_3/anaconda3/lib/python3.6/site-packages/sklearn/preprocessing/label.py:151: Deprecatio
  if diff:
/home/amd_3/anaconda3/lib/python3.6/site-packages/sklearn/preprocessing/label.py:151: Deprecatio
  if diff:
/home/amd_3/anaconda3/lib/python3.6/site-packages/sklearn/preprocessing/label.py:151: Deprecatio
  if diff:
/home/amd_3/anaconda3/lib/python3.6/site-packages/sklearn/preprocessing/label.py:151: Deprecatio
  if diff:
```

```
/home/amd_3/anaconda3/lib/python3.6/site-packages/sklearn/preprocessing/label.py:151: Deprecatio
  if diff:
/home/amd_3/anaconda3/lib/python3.6/site-packages/sklearn/preprocessing/label.py:151: Deprecatio
  if diff:
/home/amd_3/anaconda3/lib/python3.6/site-packages/sklearn/preprocessing/label.py:151: Deprecatio
  if diff:
/home/amd_3/anaconda3/lib/python3.6/site-packages/sklearn/preprocessing/label.py:151: Deprecatio
  if diff:
```

```
/home/amd_3/anaconda3/lib/python3.6/site-packages/sklearn/preprocessing/label.py:151: Deprecatio
  if diff:
/home/amd_3/anaconda3/lib/python3.6/site-packages/sklearn/preprocessing/label.py:151: Deprecatio
  if diff:
/home/amd_3/anaconda3/lib/python3.6/site-packages/sklearn/preprocessing/label.py:151: Deprecatio
  if diff:
/home/amd_3/anaconda3/lib/python3.6/site-packages/sklearn/preprocessing/label.py:151: Deprecatio
  if diff:
```

```
/home/amd_3/anaconda3/lib/python3.6/site-packages/sklearn/preprocessing/label.py:151: Deprecatio
  if diff:
/home/amd_3/anaconda3/lib/python3.6/site-packages/sklearn/preprocessing/label.py:151: Deprecatio
  if diff:
/home/amd_3/anaconda3/lib/python3.6/site-packages/sklearn/preprocessing/label.py:151: Deprecatio
  if diff:
/home/amd_3/anaconda3/lib/python3.6/site-packages/sklearn/preprocessing/label.py:151: Deprecatio
  if diff:
```

```
/home/amd_3/anaconda3/lib/python3.6/site-packages/sklearn/preprocessing/label.py:151: Deprecatio
  if diff:
/home/amd_3/anaconda3/lib/python3.6/site-packages/sklearn/preprocessing/label.py:151: Deprecatio
  if diff:
/home/amd_3/anaconda3/lib/python3.6/site-packages/sklearn/preprocessing/label.py:151: Deprecatio
  if diff:
/home/amd_3/anaconda3/lib/python3.6/site-packages/sklearn/preprocessing/label.py:151: Deprecatio
  if diff:
```

```
/home/amd_3/anaconda3/lib/python3.6/site-packages/sklearn/preprocessing/label.py:151: Deprecatio
  if diff:
/home/amd_3/anaconda3/lib/python3.6/site-packages/sklearn/preprocessing/label.py:151: Deprecatio
  if diff:
/home/amd_3/anaconda3/lib/python3.6/site-packages/sklearn/preprocessing/label.py:151: Deprecatio
  if diff:
/home/amd_3/anaconda3/lib/python3.6/site-packages/sklearn/preprocessing/label.py:151: Deprecatio
  if diff:
```

```
/home/amd_3/anaconda3/lib/python3.6/site-packages/sklearn/preprocessing/label.py:151: Deprecatio
  if diff:
/home/amd_3/anaconda3/lib/python3.6/site-packages/sklearn/preprocessing/label.py:151: Deprecatio
  if diff:
/home/amd_3/anaconda3/lib/python3.6/site-packages/sklearn/preprocessing/label.py:151: Deprecatio
  if diff:
/home/amd_3/anaconda3/lib/python3.6/site-packages/sklearn/preprocessing/label.py:151: Deprecatio
  if diff:
```

```
/home/amd_3/anaconda3/lib/python3.6/site-packages/sklearn/preprocessing/label.py:151: Deprecatio
  if diff:
/home/amd_3/anaconda3/lib/python3.6/site-packages/sklearn/preprocessing/label.py:151: Deprecatio
  if diff:
/home/amd_3/anaconda3/lib/python3.6/site-packages/sklearn/preprocessing/label.py:151: Deprecatio
  if diff:
/home/amd_3/anaconda3/lib/python3.6/site-packages/sklearn/preprocessing/label.py:151: Deprecatio
  if diff:
```

```
/home/amd_3/anaconda3/lib/python3.6/site-packages/sklearn/preprocessing/label.py:151: Deprecatio
  if diff:
/home/amd_3/anaconda3/lib/python3.6/site-packages/sklearn/preprocessing/label.py:151: Deprecatio
  if diff:
/home/amd_3/anaconda3/lib/python3.6/site-packages/sklearn/preprocessing/label.py:151: Deprecatio
  if diff:
/home/amd_3/anaconda3/lib/python3.6/site-packages/sklearn/preprocessing/label.py:151: Deprecatio
  if diff:
```

```
==========================================================================================
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++


 The depth vs AUC score plot
```

depth vs AUC
1.00
0.95
0.90
0.85
AUC Scores
0
10
20
30
40
50
depth
Train AUC
Validation AUC

```
++++++
```

num_estimators vs AUC
1.00
0.95
0.90
0.85
AUC Scores
Train AUC
Validation AUC
50
100
150
200
num_estimatores

++++++

Hyper param Heatmap DT-Train
max_depth
2
5
20
50
0.83
0.85
0.87
0.88
0.9
0.92
0.94
0.96
0.99
0.99
0.99
0.99
0.99
0.99
0.99
0.99
40
70
120
200
n_estimators
0.99
0.96
0.93
0.90
0.87
0.84

Hyper param Heatmap DT-Validation
max_depth
2
5
20
50
0.82
0.84
0.85
0.86
0.86
0.87
0.87
0.87
0.86
0.87
0.87
0.88
0.86
0.87
0.87
0.88
40
70
120
200
n_estimators
0.87
0.86
0.85
0.84
0.83

```
Best hyperparam value:  (20, 200)
```

```
/home/amd_3/anaconda3/lib/python3.6/site-packages/sklearn/preprocessing/label.py:151: Deprecatio
  if diff:
```

	Negative	Positive
Precision	0.999950	1.000000
Recall	1.000000	0.999950
Fscore	0.999975	0.999975
Support	20024.000000	19976.000000

```
/home/amd_3/anaconda3/lib/python3.6/site-packages/sklearn/preprocessing/label.py:151: Deprecatio
  if diff:
```

```
Test auc score  0.8838408303933897
```

```
              Negative      Positive
Precision     0.439014      0.953011
Recall        0.817904      0.779428
Fscore        0.571352      0.857524
Support    2614.000000  12386.000000
Results Summary:
 [('Hyper Param', (20, 200)), ('AUC', '0.8838'), ('f-score(-ve)', '57.1352'), ('f-score(+ve)', '
```

4.6 Observation on XGB set 1, set 2, set 3, set 4

XGB has more number of hyper parameters to tune and is computationally expensive to search.

All the xgb model has AUC score above 0.88

5 Procedure Summary

Train random forest with different hyper parameter setting on all 4 datasets

Identify the best hyper parameter usig cross validation method

Build final RF model by using the best hyper param identified

Visualize the performace of model using ROC curve & confusion matrix
Represent the important features identified by the decision tree in word cloud
Train XGB models with different hyper parameter setting on all 4 datasets
Identify the best hyper parameter usig cross validation method
Build final XGB model by using the best hyper param identified
Visualize the performace of model using ROC curve & confusion matrix

6 Results Summary

```
In [31]: from prettytable import PrettyTable
         Pret_table = PrettyTable()
         Pret_table.field_names = ['Vectorizer', 'Model', '(max_depth, n_estimators)',
                                   'AUC', 'Fscore (-ve)', 'Fscore (+ve)']
         Pret_table.title = 'Ensemble Results Summary'
```

```
In [32]: # RF Results Summary
         Pret_table.add_row(['BoW', 'RF'] + ptabe_entry_a1)
         Pret_table.add_row(['TF-IDF', 'RF'] + ptabe_entry_a2)
         Pret_table.add_row(['Avg W2V', 'RF'] + ptabe_entry_a3)
         Pret_table.add_row(['TF-IDF W2V', 'RF'] + ptabe_entry_a4)

         # XGBoost Results Summary
         Pret_table.add_row(['BoW', 'XGB'] + ptabe_entry_b1)
         Pret_table.add_row(['TF-IDF', 'XGB'] + ptabe_entry_b2)
         Pret_table.add_row(['Avg W2V', 'XGB'] + ptabe_entry_b3)
         Pret_table.add_row(['TF-IDF W2V', 'XGB'] + ptabe_entry_b4)
```

```
In [33]: print(Pret_table)
```

```
+-------------------------------------------------------------------------------------+
|                                Ensemble Results Summary                             |
+------------+-------+--------------------------+--------+--------------+--------------+
| Vectorizer | Model | (max_depth, n_estimators) |  AUC   | Fscore (-ve) | Fscore (+ve) |
+------------+-------+--------------------------+--------+--------------+--------------+
|    BoW     |   RF  |         (50, 500)        | 0.9172 |   62.8417    |   88.2320    |
|   TF-IDF   |   RF  |         (50, 300)        | 0.9148 |   62.7832    |   88.2811    |
|  Avg W2V   |   RF  |         (10, 500)        | 0.9112 |   61.7723    |   87.8103    |
| TF-IDF W2V |   RF  |         (10, 500)        | 0.8611 |   54.4582    |   84.3995    |
|    BoW     |  XGB  |         (20, 200)        | 0.9324 |   65.6725    |   89.6031    |
|   TF-IDF   |  XGB  |         (20, 200)        | 0.9305 |   65.3120    |   89.4664    |
|  Avg W2V   |  XGB  |          (5, 200)        | 0.9251 |   64.3793    |   89.1104    |
| TF-IDF W2V |  XGB  |         (20, 200)        | 0.8838 |   57.1352    |   85.7524    |
+------------+-------+--------------------------+--------+--------------+--------------+
```

7 Conclusions

XGB models outperformed when compared with RF models. The f-score for +ve class as well as -ve class is higher when compared with RF model

The f-score of XGB model for -ve class is approximately 3% more than the f-score of RF model for -ve class