```
from google.colab import drive
drive.mount('/content/drive')
```

```
    Mounted at /content/drive
```

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
import re
import nltk

from sklearn.model_selection import train_test_split
from sklearn import metrics
from sklearn.svm import SVC
df1 = pd.read_csv('/content/drive/MyDrive/ML Project/Mental-Health-Twitter.csv', index_col=[0])
df2 = pd.read_csv('/content/drive/MyDrive/ML Project/sentiment_tweets3.csv', index_col=[0])
df3 = pd.read_excel('/content/drive/MyDrive/ML Project/dataset.xlsx', index_col=[0])
```

```
df2
df1
```

	post_id	post_created	post_text	user_id	followers	friends	favourites	sta
0	637894677824413696	Sun Aug 30 07:48:37 +0000 2015	It's just over 2 years since I was diagnosed w...	1013187241	84	211	251	
1	637890384576778240	Sun Aug 30 07:31:33 +0000 2015	It's Sunday, I need a break, so I'm planning t...	1013187241	84	211	251	
2	637749345908051968	Sat Aug 29 22:11:07 +0000 2015	Awake but tired. I need to sleep but my brain ...	1013187241	84	211	251	

```
df2.head()
```

	post_text	label
Index		
106	just had a real good moment. i missssssssss hi...	0
217	is reading manga http://plurk.com/p/mzp1e	0
220	@comeagainjen http://twitpic.com/2y2lx - http:...	0
288	@lapcat Need to send 'em to my accountant tomo...	0
540	ADD ME ON MYSPACE!!! myspace.com/LookThunder	0

```
df3
```

	label
text	
oh my gosh	1.0
trouble sleeping, confused mind, restless heart. All out of tune	1.0
All wrong, back off dear, forward doubt. Stay in a restless and restless place	1.0
I've shifted my focus to something else but I'm still worried	1.0
I'm restless and restless, it's been a month now, boy. What do you mean?	1.0
...	...
I can't forget you #SpiritHadrian	0.0
€ ®šæœŸâ˜†ã€'..DJ DAIKI! DJ DAIKI! DJ DAIKI!.DJ DAIKI! DJ DAIKI!!!..#Hey!Say!JUMP.#æœ‰å²¡ ²´	0.0
Dai5y! <3	0.0
tired of clowns but still hopefully tonight if not tomorrow ™ but mas tm also no teaser yet.....	0.0
MW SUBI WN LA VACA	0.0

6982 rows × 1 columns

```
df1.info
```

```
<bound method DataFrame.info of                   post_id                     post_created  \
0      637894677824413696  Sun Aug 30 07:48:37 +0000 2015
1      637890384576778240  Sun Aug 30 07:31:33 +0000 2015
2      637749345908051968  Sat Aug 29 22:11:07 +0000 2015
3      637696421077123073  Sat Aug 29 18:40:49 +0000 2015
4      637696327485366272  Sat Aug 29 18:40:26 +0000 2015
...                   ...                             ...
19995  819336825231773698  Thu Jan 12 00:14:56 +0000 2017
19996  819334654260080640  Thu Jan 12 00:06:18 +0000 2017
19997  819334503042871297  Thu Jan 12 00:05:42 +0000 2017
19998  819334419374899200  Thu Jan 12 00:05:22 +0000 2017
19999  819334270825197568  Thu Jan 12 00:04:47 +0000 2017

                                               post_text     user_id  \
0      It's just over 2 years since I was diagnosed w...  1013187241
1      It's Sunday, I need a break, so I'm planning t...  1013187241
2      Awake but tired. I need to sleep but my brain ...  1013187241
3      RT @SewHQ: #Retro bears make perfect gifts and...  1013187241
4      It’s hard to say whether packing lists are mak...  1013187241
...                                                  ...         ...
19995              A day without sunshine is like night.  1169875706
19996  Boren's Laws: (1) When in charge, ponder. (2) ...  1169875706
19997  The flow chart is a most thoroughly oversold p...  1169875706
19998  Ships are safe in harbor, but they were never ...  1169875706
19999     Black holes are where God is dividing by zero.  1169875706

       followers  friends  favourites  statuses  retweets  label
0             84      211         251       837         0      1
1             84      211         251       837         1      1
2             84      211         251       837         0      1
3             84      211         251       837         2      1
4             84      211         251       837         1      1
...          ...      ...         ...       ...       ...    ...
19995        442      230           7   1063601         0      0
19996        442      230           7   1063601         0      0
19997        442      230           7   1063601         0      0
19998        442      230           7   1063601         0      0
19999        442      230           7   1063601         0      0

[20000 rows x 10 columns]>
```

```
df2.info
```

```
<bound method DataFrame.info of                                         message to examine  \
Index
106     just had a real good moment. i missssssssss hi...
217            is reading manga  http://plurk.com/p/mzp1e
220     @comeagainjen http://twitpic.com/2y2lx - http:...
288     @lapcat Need to send 'em to my accountant tomo...
540         ADD ME ON MYSPACE!!!  myspace.com/LookThunder
...                                                   ...
802309  No Depression by G Herbo is my mood from now o...
802310  What do you do when depression succumbs the br...
802311  Ketamine Nasal Spray Shows Promise Against Dep...
802312  dont mistake a bad day with depression! everyo...
802313                                                  0

        label (depression result)
Index
106                             0
217                             0
220                             0
288                             0
540                             0
...                           ...
802309                          1
802310                          1
802311                          1
802312                          1
802313                          1

[10314 rows x 2 columns]>
```

```
df3.info
```

```
<bound method DataFrame.info of                                                      label
text
oh my gosh                                             1.0
trouble sleeping, confused mind, restless heart...     1.0
All wrong, back off dear, forward doubt. Stay i...     1.0
I've shifted my focus to something else but I'm...     1.0
I'm restless and restless, it's been a month no...     1.0
...                                                    ...
I can't forget you #SpiritHadrian                      0.0
€ ®šæœŸâ˜†ã€'..DJ DAIKI! DJ DAIKI! DJ DAIKI!.DJ...     0.0
Dai5y! <3                                              0.0
tired of clowns but still hopefully tonight if ...     0.0
MW SUBI WN LA VACA                                     0.0

[6982 rows x 1 columns]>
```

```
df1.isnull().all()
```

```
post_id         False
post_created    False
post_text       False
user_id         False
followers       False
friends         False
```

```
     favourites      False
     statuses        False
     retweets        False
     label           False
     dtype: bool
```

```
df2.isnull().all()
```

```
     message to examine            False
     label (depression result)     False
     dtype: bool
```

```
df3.isnull().all()
```

```
     label    False
     dtype: bool
```

```
# specify columns to keep for d2
df1 = df1[['post_text', 'label']]

# confirm columns kept are the correct ones we want
df1.head()
```

	post_text	label
0	It's just over 2 years since I was diagnosed w...	1
1	It's Sunday, I need a break, so I'm planning t...	1
2	Awake but tired. I need to sleep but my brain ...	1
3	RT @SewHQ: #Retro bears make perfect gifts and...	1
4	It's hard to say whether packing lists are mak...	1

```
# standardize column names for df2
df2.columns= ['post_text', 'label']
df2.head()
```

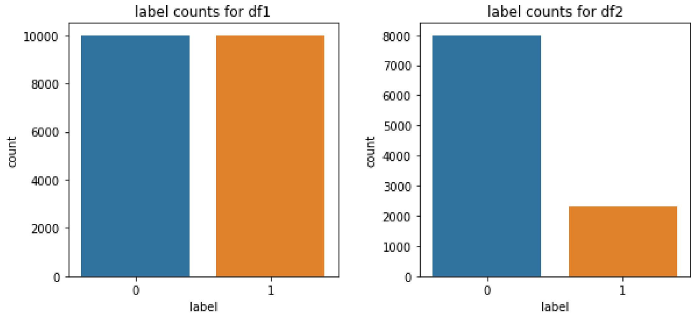
	post_text	label
Index		
106	just had a real good moment. i missssssssss hi...	0
217	is reading manga http://plurk.com/p/mzp1e	0
220	@comeagainjen http://twitpic.com/2y2lx - http:...	0
288	@lapcat Need to send 'em to my accountant tomo...	0
540	ADD ME ON MYSPACE!!! myspace.com/LookThunder	0

```
# set up space for figure
fig = plt.figure(figsize=(15,4))
fig.subplots_adjust(hspace=0.3, wspace=0.3)

# counts for df1
ax1 = fig.add_subplot(1,3,2)
sns.countplot(x='label', data=df1)
ax1.set(title="label counts for df1")

# counts for df2
ax2 = fig.add_subplot(1,3,3)
sns.countplot(x='label', data=df2)
ax2.set(title="label counts for df2")
```

```
     [Text(0.5, 1.0, 'label counts for df2')]
```



```
# look at actual numbers for df1
df1['label'].value_counts()
```

```
     1    10000
     0    10000
     Name: label, dtype: int64
```

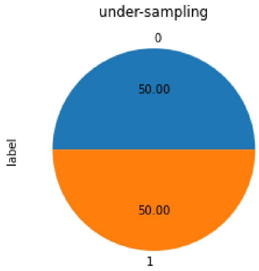
```
# look at actual numbers for df2
df2['label'].value_counts()
```

```
0    8000
1    2314
Name: label, dtype: int64
```

```
from imblearn.under_sampling import RandomUnderSampler
X = df2.drop(['label'], axis=1)
y = df2['label']

rus = RandomUnderSampler(sampling_strategy='not minority')
X_res, y_res = rus.fit_resample(X, y)

# visualize proportion of labels after balancing
ax = y_res.value_counts().plot.pie(autopct='%.2f')
_ = ax.set_title("under-sampling")
```



```
# class distribution
y_res.value_counts()
```

```
0    2314
1    2314
Name: label, dtype: int64
```

```
# combine columns for d3
df2 = pd.concat([X_res, y_res], axis='columns')

# confirm successful concatenation
df2.head()

# confirm balanced data
df2['label'].value_counts()
```

```
0    2314
1    2314
Name: label, dtype: int64
```

```
df = pd.concat([df1, df2])

# confirm change
df.shape
```

```
(24628, 2)
```

```
# split words in 'post-text' and count
df['word_count'] = df['post_text'].apply(lambda x: len(str(x).split()))
df.head()
```

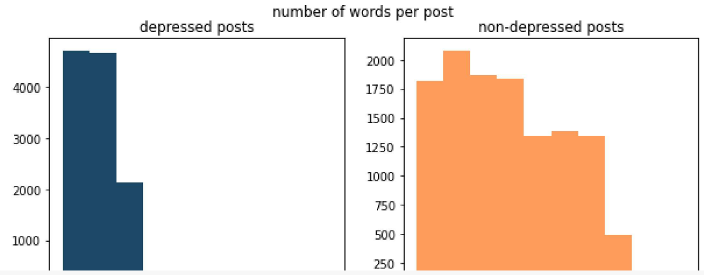
	post_text	label	word_count
0	It's just over 2 years since I was diagnosed w...	1	26
1	It's Sunday, I need a break, so I'm planning t...	1	19
2	Awake but tired. I need to sleep but my brain ...	1	13
3	RT @SewHQ: #Retro bears make perfect gifts and...	1	23
4	It's hard to say whether packing lists are mak...	1	21

```
# plot number of words in 'depressed' and 'non-depressed' posts
fig,(ax1, ax2) = plt.subplots(1, 2, figsize = (10, 4))

dep_words = df[df['label'] == 1]['word_count']
ax1.hist(dep_words,color='#1c4966')
ax1.set_title('depressed posts')

non_dep_words = df[df['label'] == 0]['word_count']
ax2.hist(non_dep_words, color = '#ff9d5c')
ax2.set_title('non-depressed posts')

fig.suptitle('number of words per post')
plt.show()
```



```
print('The minimum number of words in non-depressed posts is: ', non_dep_words.min())
print('The maximum number of words in non-depressed posts is: ', non_dep_words.max())
print('____________________________________________________________')
print('The minimum number of words in depressed posts is: ', dep_words.min())
print('The maximum number of words in non-depressed posts is: ', dep_words.max())
print('____________________________________________________________')
print('The average number of words in non-depressed posts is: ', round(non_dep_words.mean()))
print('The average number of words in depressed posts is: ', round(dep_words.mean()))
```

```
The minimum number of words in non-depressed posts is:  1
The maximum number of words in non-depressed posts is:  34
____________________________________________________________
The minimum number of words in depressed posts is:  1
The maximum number of words in non-depressed posts is:  92
____________________________________________________________
The average number of words in non-depressed posts is:  12
The average number of words in depressed posts is:  15
```

Now we are splitting data into training and testing data set.

```
from sklearn.model_selection import train_test_split

# split into training and testing site with 20% of rows going to testing and 80% going to training
# random state of 10 ensures reproducibility
X_train, X_test, y_train, y_test = train_test_split(df['post_text'], df['label'], test_size = 0.3, random_state = 10)

print(f"Number of training: {X_train.shape[0]}")
print(f"Number of testing: {X_test.shape[0]}")
```

```
Number of training: 17239
Number of testing: 7389
```

```
from bs4 import BeautifulSoup

# create stepwise cleaning function
def preprocess(text):

    # lowercase text
    text = text.lower()

    # strip all excess white space
    text = text.strip()

    # strip HTML tags
    text = BeautifulSoup(text, 'html.parser').get_text(separator = ' ')

    # remove retweets
    text = re.sub('rt @[\w_]+:','', text)

    # remove hyperlinks
    text = re.sub(r'http\S+','', text)

    # remove escape sequences
    text = re.sub(r'\n','', text)

    # remove punctuations
    text = re.sub(r'[^A-Za-z0-9]+', ' ', text)

    return text
```

```
import nltk
nltk.download('stopwords')
```

```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Unzipping corpora/stopwords.zip.
True
```

```
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize

stop_words = set(stopwords.words('english'))

# remove stopwords
def remove_stopwords(text):
    word_tokens = word_tokenize(text)
    text_no_stop = [word for word in word_tokens if word not in stop_words]
    return text_no_stop
```

```
from nltk.stem import WordNetLemmatizer

# lemmatize
def lemmatize_text(list_of_tokenized_words):
    lemmatizer = WordNetLemmatizer()
    lemmatized = [lemmatizer.lemmatize(token) for token in list_of_tokenized_words]
    return lemmatized
```

```
# put it all together
def final_preprocess(text):
    return preprocess(remove_stopwords(lemmatize_text(text)))

# pre-process training data
X_train_clean = [preprocess(text) for text in X_train]
```

For Running any machine learning models, text data must be converted into numerical feauture vectors, so that machine can learn the code easily and consume less time.

```
from sklearn.feature_extraction.text import TfidfVectorizer

# initialize
tfidf_vectorizer = TfidfVectorizer(min_df = 0.0003)
tfidf_vec_matrix = tfidf_vectorizer.fit_transform(X_train_clean)
feature_names = tfidf_vectorizer.get_feature_names_out()

# create dense matrix and convert to dataframe
dense_mtx = tfidf_vec_matrix.todense()
dense_lst = dense_mtx.tolist()
tfidf_df = pd.DataFrame(dense_lst, columns = feature_names)
tfidf_df.head()
```

	00	000	08	10	100	1000	101	11	12	13	...	youtube	yr	yrs	yummy	yup	zayin	zayn	za
0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
4	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	

5 rows × 3524 columns

Now we have to Train the classifier.

```
from sklearn.naive_bayes import MultinomialNB

# implement Multinomial Naive Bayes algorithm for classfication
mnb_classifier = MultinomialNB().fit(tfidf_vec_matrix, y_train)
```

```
from sklearn import svm

# implement SVM algorithm for classification
svm_classifier = svm.SVC(kernel = 'rbf').fit(tfidf_vec_matrix, y_train)
```

```
from sklearn.linear_model import LogisticRegression

# implement Logistic Regression algorithm for classification
lr_classifier = LogisticRegression().fit(tfidf_vec_matrix, y_train)
```

```
from sklearn.neighbors import KNeighborsClassifier
knn_classifier = KNeighborsClassifier().fit(tfidf_vec_matrix, y_train)
```

Evaluating Performance.

The effectiveness of a classification model is assessed using a confusion matrix, which compares the proportion of properly and incorrectly labelled predictions. It is divided into four quadrants: True Positive (properly labelled positive class), Fake Positive (incorrectly labelled positive class), True Negative (properly labelled negative class), and False Negative (incorrectly labelled negative class).

The percentage of properly labelled predictions is known as accuracy.

The percentage of true positives among all favourable predictions is known as precision.

Recall is the percentage of genuine positive labels that are also true positives.

The weighted average of recall and precision is the F1 number.

We can determine what kinds of mistakes came from our models by looking at our confusion matrices.

```
from sklearn import metrics
from sklearn.metrics import confusion_matrix
from sklearn.metrics import plot_confusion_matrix
```

```
# preprocess testing data
X_test_clean = [preprocess(text) for text in X_test]

# vectorize testing data
X_test_tfidf_vec_matrix = tfidf_vectorizer.transform(X_test_clean)
```

Multinomial Naive Bayes.

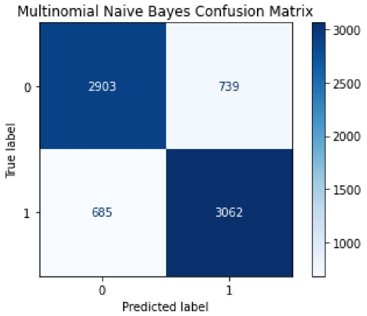
```
# predict with Naive Bayes
mnb_y_pred = mnb_classifier.predict(X_test_tfidf_vec_matrix)

# run confusion marix
confusion_matrix(y_test, mnb_y_pred)

# plot confusion matrix
mnb_confusion = plot_confusion_matrix(mnb_classifier, X_test_tfidf_vec_matrix, y_test, cmap = plt.cm.Blues)
mnb_confusion.ax_.set_title('Multinomial Naive Bayes Confusion Matrix')
plt.show()

# evaluate NB performance
print('Accuracy:', metrics.accuracy_score(y_test, mnb_y_pred))
print('Precision:', metrics.precision_score(y_test, mnb_y_pred))
print('Recall:', metrics.recall_score(y_test, mnb_y_pred))
print('F1 score:', metrics.f1_score(y_test, mnb_y_pred))
```

```
/usr/local/lib/python3.8/dist-packages/sklearn/utils/deprecation.py:87: FutureWarning: Function plc
  warnings.warn(msg, category=FutureWarning)
```



```
Accuracy: 0.8072810935173907
Precision: 0.8055774796106288
Recall: 0.8171870829997331
F1 score: 0.8113407525172229
```

Support vector Machine.

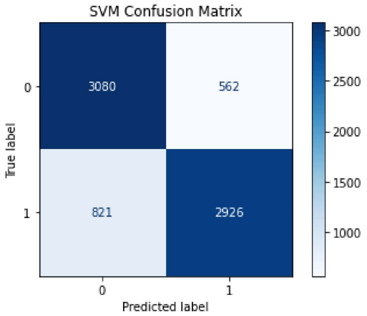
```
# predict with SVM
svm_y_pred = svm_classifier.predict(X_test_tfidf_vec_matrix)

# run confusion marix
confusion_matrix(y_test, svm_y_pred)

# plot confusion matrix
svm_confusion = plot_confusion_matrix(svm_classifier, X_test_tfidf_vec_matrix, y_test, cmap = plt.cm.Blues)
svm_confusion.ax_.set_title('SVM Confusion Matrix')
plt.show()

# evaluate SVM performance
print('Accuracy:', metrics.accuracy_score(y_test, svm_y_pred))
print('Precision:', metrics.precision_score(y_test, svm_y_pred))
print('Recall:', metrics.recall_score(y_test, svm_y_pred))
print('F1 score:', metrics.f1_score(y_test, svm_y_pred))
```

```
/usr/local/lib/python3.8/dist-packages/sklearn/utils/deprecation.py:87: FutureWarning: Function plc
  warnings.warn(msg, category=FutureWarning)
```



```
Accuracy: 0.8128298822574097
Precision: 0.8388761467889908
Recall: 0.7808913797704831
F1 score: 0.8088458880442295
```

Logistic Regression.

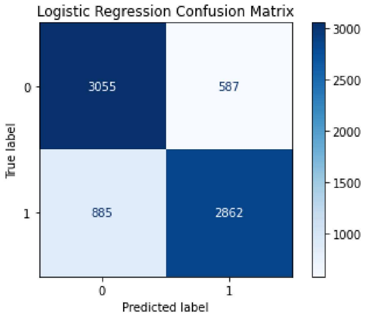
```
# predict with LR
lr_y_pred = lr_classifier.predict(X_test_tfidf_vec_matrix)

# run confusion marix
confusion_matrix(y_test, lr_y_pred)

# plot confusion matrix
lr_confusion = plot_confusion_matrix(lr_classifier, X_test_tfidf_vec_matrix, y_test, cmap = plt.cm.Blues)
lr_confusion.ax_.set_title('Logistic Regression Confusion Matrix')
plt.show()

# evaluate LR performance
print('Accuracy:', metrics.accuracy_score(y_test, lr_y_pred))
print('Precision:', metrics.precision_score(y_test, lr_y_pred))
print('Recall:', metrics.recall_score(y_test, lr_y_pred))
print('F1 score:', metrics.f1_score(y_test, lr_y_pred))
```

```
/usr/local/lib/python3.8/dist-packages/sklearn/utils/deprecation.py:87: FutureWarning: Function plc
  warnings.warn(msg, category=FutureWarning)
```



```
Accuracy: 0.8007849506022466
Precision: 0.8298057407944331
Recall: 0.7638110488390712
F1 score: 0.7954419121734297
```

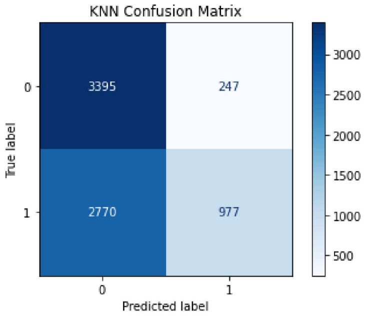
```
# predict with KNN
knn_y_pred = knn_classifier.predict(X_test_tfidf_vec_matrix)

# run confusion marix
confusion_matrix(y_test, knn_y_pred)

# plot confusion matrix
knn_confusion = plot_confusion_matrix(knn_classifier, X_test_tfidf_vec_matrix, y_test, cmap = plt.cm.Blues)
knn_confusion.ax_.set_title('KNN Confusion Matrix')
plt.show()

# evaluate KNN performance
print('Accuracy:', metrics.accuracy_score(y_test, knn_y_pred))
print('Precision:', metrics.precision_score(y_test, knn_y_pred))
print('Recall:', metrics.recall_score(y_test, knn_y_pred))
print('F1 score:', metrics.f1_score(y_test, knn_y_pred))
```

```
/usr/local/lib/python3.8/dist-packages/sklearn/utils/deprecation.py:87: FutureWarning: Function plc
  warnings.warn(msg, category=FutureWarning)
```



```
Accuracy: 0.5916903505210448
Precision: 0.798202614379085
Recall: 0.2607419268748332
F1 score: 0.3930798632065983
```

```
user = input("Enter a Text: ")
data = tfidf_vectorizer.transform([user]).toarray()
output = mnb_classifier.predict(data)
print(output)
```

```
Enter a Text: happy
[0]
```

```
user = input("Enter a Text: ")
data = tfidf_vectorizer.transform([user]).toarray()
data = tfidf_vectorizer.transform([user]).toarray()
output = svm_classifier.predict(data)
print(output)
```

```
Enter a Text: lonely
[1]
```

```
user = input("Enter a Text: ")
data = tfidf_vectorizer.transform([user]).toarray()
output = lr_classifier.predict(data)
print(output)
```

```
Enter a Text: I am feeling confused and lonely right now.
[1]
```

✓ 5s completed at 10:33 AM