Models_kNN_LDA_QDA_DT_RF

Group 4: Kiet Ly, Mary Monroe, and Shaswati Mukherjee

In [1]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

from sklearn.model_selection import cross_val_score, train_test_split
from sklearn.utils import resample
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import AdaBoostClassifier
from sklearn.metrics import accuracy_score,r2_score
from sklearn.neighbors import KNeighborsClassifier
from sklearn.preprocessing import scale
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from sklearn.discriminant_analysis import QuadraticDiscriminantAnalysis

%matplotlib inline

import seaborn as sns
sns.set(style='whitegrid')
pd.set_option('display.width', 1500)
pd.set_option('display.max_columns', 100)

import warnings
warnings.filterwarnings('ignore')
```

Create train/test set for accounts and tweets

Discussion:

The dataset for accounts and tweets are split into train/test ratio of 2/3 and 1/3. From Milestone#3, we have determined only these features are important.

Base Tweets Features
retweet_count
favorite_count
num_urls
num_mentions
num_hashtags

In [2]:

```
combine_df = []
for file_ in ['../../data/tweets_nlp_1_2_ld.csv','../../data/tweets_nlp_2_2_ld.c
sv','../../data/tweets_nlp_3_2_ld.csv']:
    df = pd.read_csv(file_,index_col=None, header=0,keep_default_na=False)
    combine_df.append(df)
all_tweets = pd.concat(combine_df, axis = 0, ignore_index = True)
all_tweets_df = all_tweets[['retweet_count', 'favorite_count', 'num_hashtags', '
num_urls', 'num_mentions','user_type']]
train_base_tweets_df, test_base_tweets_df = train_test_split(all_tweets_df, test
_size=0.33, random_state=42, stratify=all_tweets_df['user_type'])
display(train_base_tweets_df.head(2))
print('train tweets shape:',train_base_tweets_df.shape)
print('test tweets shape:',test_base_tweets_df.shape)
```

	retweet_count	favorite_count	num_hashtags	num_urls	num_mentions	user_type
54933	0	0	0	1	0	0
80644	0	0	0	1	0	0

```
train tweets shape: (80574, 6)
test tweets shape: (39686, 6)
```

In [3]:

```
all_tweets_df = all_tweets[['retweet_count', 'favorite_count', 'num_hashtags', '
num_urls', 'num_mentions',
                               'user_type', 'sentiment_negative', 'sentiment_neu
tral', 'sentiment_positive',
                               'ratio_pos', 'ratio_neg', 'ratio_neu', 'token_cou
nt', 'url_token_ratio', 'ant',
                               'disgust', 'fear', 'joy', 'sadness', 'surprise',
'trust','jaccard']]
all_tweets_df.head(2)
```

Out[3]:

	retweet_count	favorite_count	num_hashtags	num_urls	num_mentions	user_type	sentimer
0	0	0	0	1	0	1	
1	0	0	0	1	1	1	

In [4]:

```
all_tweets_df = all_tweets[['retweet_count', 'favorite_count', 'num_hashtags', '
num_urls', 'num_mentions',
                               'user_type', 'sentiment_negative', 'sentiment_neu
tral', 'sentiment_positive',
                               'ratio_pos', 'ratio_neg', 'ratio_neu', 'token_cou
nt', 'url_token_ratio', 'ant',
                               'disgust', 'fear', 'joy', 'sadness', 'surprise',
'trust','jaccard','LD-uber_index','LD-yule_s_k','LD-mtld','LD-hdd']]

all_tweets_df.head(2)
```

Out[4]:

	retweet_count	favorite_count	num_hashtags	num_urls	num_mentions	user_type	sentimer
0	0	0	0	1	0	1	
1	0	0	0	1	1	1	

kNN without NLP

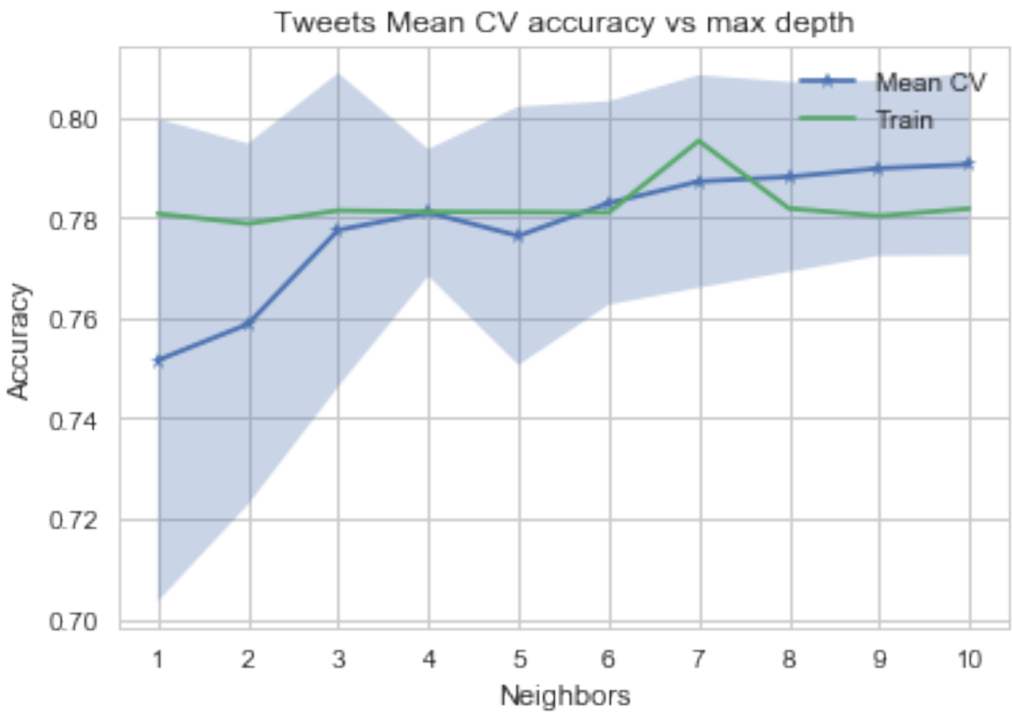
In [5]:

```
#kNN or take forever.
all_tweets_df = all_tweets[['retweet_count', 'favorite_count', 'num_hashtags', '
num_urls', 'num_mentions','user_type']].sample(frac=.30)
train_base_tweets_df, test_base_tweets_df = train_test_split(all_tweets_df, test
_size=0.33, random_state=42, stratify=all_tweets_df['user_type'])

X_train, y_train = train_base_tweets_df.drop('user_type',axis=1), train_base_twe
ets_df['user_type']
X_test, y_test = test_base_tweets_df.drop('user_type',axis=1), test_base_tweets_
df['user_type']
Xs_train, Xs_test = scale(X_train), scale(X_test)

neighbors, train_scores, cvmeans, cvstds, cv_scores = [], [], [], [], []
for n in range(1,11):
    neighbors.append(n)
    knn = KNeighborsClassifier(n_neighbors = n)
    train_scores.append(knn.fit(X_train, y_train).score(X_train, y_train))
    scores = cross_val_score(estimator=knn,X=Xs_train, y=y_train, cv=5)
    cvmeans.append(scores.mean())
    cvstds.append(scores.std())

#Alter data structure for using internal numpy functions
cvmeans = np.array(cvmeans)
cvstds = np.array(cvstds)
#Plot Means and Shade the +-2 SD Interval
plt.plot(neighbors, cvmeans, '*-', label="Mean CV")
plt.fill_between(neighbors, cvmeans - 2*cvstds, cvmeans + 2*cvstds, alpha=0.3)
ylim = plt.ylim()
plt.plot(neighbors, train_scores, '-+', label="Train")
plt.ylim(ylim)
plt.legend()
plt.ylabel("Accuracy")
plt.xlabel("Neighbors")
plt.title('Tweets Mean CV accuracy vs max depth')
plt.xticks(neighbors)
plt.show()
```


Tweets Mean CV accuracy vs max depth
Mean CV
Train
0.80
0.78
0.76
0.74
0.72
0.70
Accuracy
1
2
3
4
5
6
7
8
9
10
Neighbors

kNN with NLP

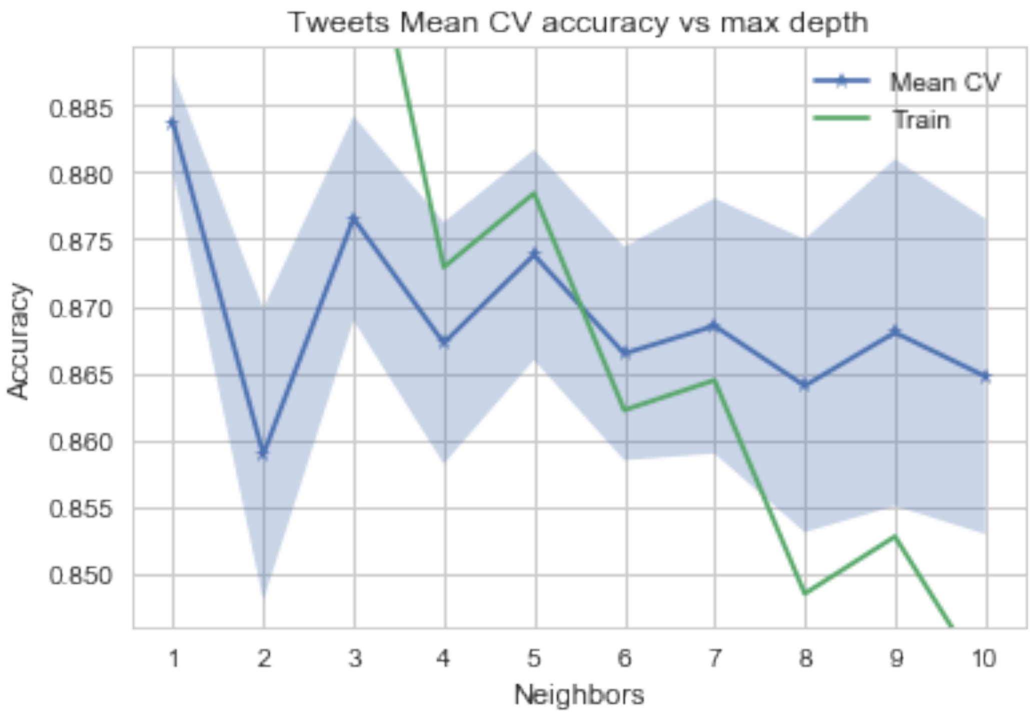
In [6]:

```
#scale down to 10% or take forever.
all_tweets_df = all_tweets[['retweet_count', 'favorite_count', 'num_hashtags', '
num_urls', 'num_mentions',
                              'user_type', 'sentiment_negative', 'sentiment_neu
tral', 'sentiment_positive',
                              'ratio_pos', 'ratio_neg', 'ratio_neu', 'token_cou
nt', 'url_token_ratio', 'ant',
                              'disgust', 'fear', 'joy', 'sadness', 'surprise',
'trust','jaccard']].sample(frac=.30)
train_base_tweets_df, test_base_tweets_df = train_test_split(all_tweets_df, test
_size=0.33, random_state=42, stratify=all_tweets_df['user_type'])

X_train, y_train = train_base_tweets_df.drop('user_type',axis=1), train_base_twe
ets_df['user_type']
X_test, y_test = test_base_tweets_df.drop('user_type',axis=1), test_base_tweets_
df['user_type']
Xs_train, Xs_test = scale(X_train), scale(X_test)

neighbors, train_scores, cvmeans, cvstds, cv_scores = [], [], [], [], []
for n in range(1,11):
    neighbors.append(n)
    knn = KNeighborsClassifier(n_neighbors = n)
    train_scores.append(knn.fit(X_train, y_train).score(X_train, y_train))
    scores = cross_val_score(estimator=knn,X=Xs_train, y=y_train, cv=5)
    cvmeans.append(scores.mean())
    cvstds.append(scores.std())

#Alter data structure for using internal numpy functions
cvmeans = np.array(cvmeans)
cvstds = np.array(cvstds)
#Plot Means and Shade the +-2 SD Interval
plt.plot(neighbors, cvmeans, '*-', label="Mean CV")
plt.fill_between(neighbors, cvmeans - 2*cvstds, cvmeans + 2*cvstds, alpha=0.3)
ylim = plt.ylim()
plt.plot(neighbors, train_scores, '-+', label="Train")
plt.ylim(ylim)
plt.legend()
plt.ylabel("Accuracy")
plt.xlabel("Neighbors")
plt.title('Tweets Mean CV accuracy vs max depth')
plt.xticks(neighbors)
plt.show()
```



LDA/QDA without NLP

In [7]:

```
all_tweets_df = all_tweets[['retweet_count', 'favorite_count', 'num_hashtags', '
num_urls', 'num_mentions','user_type']]
train_base_tweets_df, test_base_tweets_df = train_test_split(all_tweets_df, test
_size=0.33, random_state=42, stratify=all_tweets_df['user_type'])

X_train, y_train = train_base_tweets_df.drop('user_type',axis=1), train_base_twe
ets_df['user_type']
X_test, y_test = test_base_tweets_df.drop('user_type',axis=1), test_base_tweets_
df['user_type']

lda = LinearDiscriminantAnalysis().fit(X_train, y_train)
qda = QuadraticDiscriminantAnalysis().fit(X_train, y_train)
print("LDA score: %f, CV score: %f" % (accuracy_score(y_test, lda.predict(X_test
)), cross_val_score(estimator=lda, X=X_test, y=y_test, cv=5).mean()))
print("QDA score: %f, CV score: %f" % (accuracy_score(y_test, qda.predict(X_test
)), cross_val_score(estimator=qda, X=X_test, y=y_test, cv=5).mean()))
```

```
LDA score: 0.715240, CV score: 0.716122
QDA score: 0.753943, CV score: 0.754397
```

LDA/QDA with NLP

In [8]:

```
#scale down to 10% or take forever.
all_tweets_df = all_tweets[['retweet_count', 'favorite_count', 'num_hashtags', '
num_urls', 'num_mentions',
                                'user_type', 'sentiment_negative', 'sentiment_neu
tral', 'sentiment_positive',
                                'ratio_pos', 'ratio_neg', 'ratio_neu', 'token_cou
nt', 'url_token_ratio', 'ant',
                                'disgust', 'fear', 'joy', 'sadness', 'surprise',
'trust','jaccard']]
train_base_tweets_df, test_base_tweets_df = train_test_split(all_tweets_df, test
_size=0.33, random_state=42, stratify=all_tweets_df['user_type'])

X_train, y_train = train_base_tweets_df.drop('user_type',axis=1), train_base_twe
ets_df['user_type']
X_test, y_test = test_base_tweets_df.drop('user_type',axis=1), test_base_tweets_
df['user_type']

lda = LinearDiscriminantAnalysis().fit(X_train, y_train)
qda = QuadraticDiscriminantAnalysis().fit(X_train, y_train)
print("LDA score: %f, CV score: %f" % (accuracy_score(y_test, lda.predict(X_test
)), cross_val_score(estimator=lda, X=X_test, y=y_test, cv=5).mean()))
print("QDA score: %f, CV score: %f" % (accuracy_score(y_test, qda.predict(X_test
)), cross_val_score(estimator=qda, X=X_test, y=y_test, cv=5).mean()))
```

```
LDA score: 0.810487, CV score: 0.810311
QDA score: 0.762057, CV score: 0.763493
```

Base Decision Tree without NLP

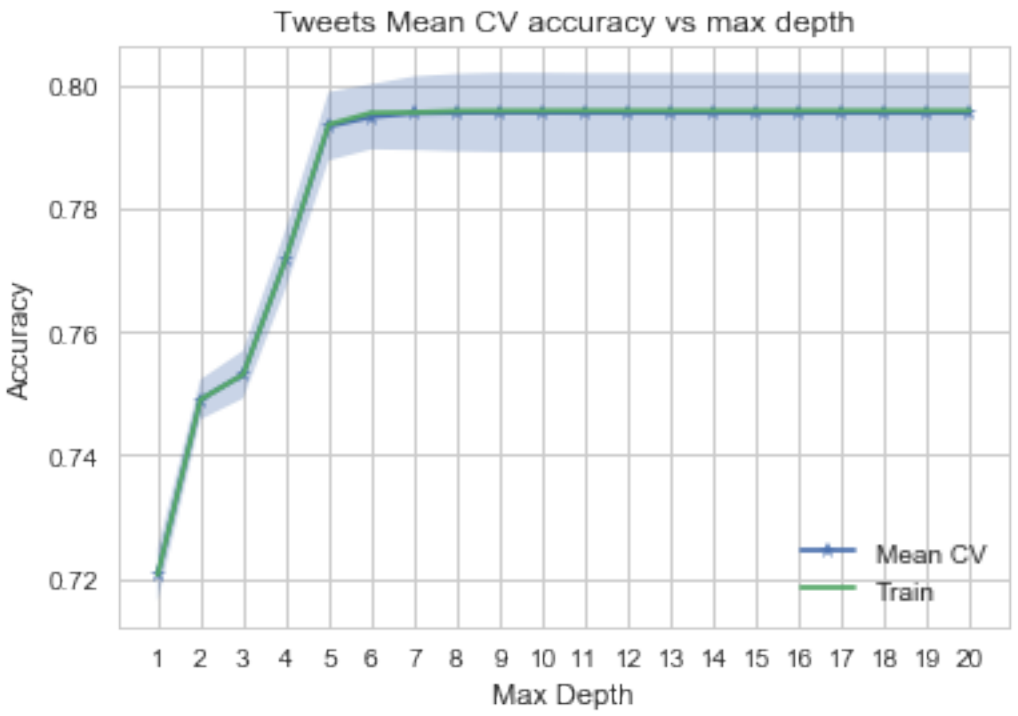
First we determine the optimal depth for Decision Tree, then use that depth to train Random Forest. From the plot, the depth > 6 does not improve accuracy. We will pick depth = 6 as best depth. The test accurracy is 0.793 and training accuracy 0.795 so nearly match, we can conlude there is no overfit issue.

In [9]:

```
#Perform 5-fold cross validation and store results
all_tweets_df = all_tweets[['retweet_count', 'favorite_count', 'num_hashtags', '
num_urls', 'num_mentions','user_type']]
train_base_tweets_df, test_base_tweets_df = train_test_split(all_tweets_df, test
_size=0.33, random_state=42,
                                                             stratify=all_tweets
_df['user_type'])
X_train, y_train = train_base_tweets_df.drop('user_type',axis=1), train_base_twe
ets_df['user_type']
X_test, y_test = test_base_tweets_df.drop('user_type',axis=1), test_base_tweets_
df['user_type']

depths, train_scores, cvmeans, cvstds, cv_scores = [], [], [], [], []
for depth in range(1,21):
    depths.append(depth)
    dt = DecisionTreeClassifier(max_depth=depth)
    train_scores.append(dt.fit(X_train, y_train).score(X_train, y_train))
    scores = cross_val_score(estimator=dt, X=X_train, y=y_train, cv=5)
    cvmeans.append(scores.mean())
    cvstds.append(scores.std())

#Alter data structure for using internal numpy functions
cvmeans = np.array(cvmeans)
cvstds = np.array(cvstds)
#Plot Means and Shade the +-2 SD Interval
plt.plot(depths, cvmeans, '*-', label="Mean CV")
plt.fill_between(depths, cvmeans - 2*cvstds, cvmeans + 2*cvstds, alpha=0.3)
ylim = plt.ylim()
plt.plot(depths, train_scores, '-+', label="Train")
plt.ylim(ylim)
plt.legend()
plt.ylabel("Accuracy")
plt.xlabel("Max Depth")
plt.title('Tweets Mean CV accuracy vs max depth')
plt.xticks(depths)
plt.show()
```



In [10]:

```
#Choosing the best depth
idx = depths.index(7)
print("Accuracy: Mean={:.3f}, +/- 2 SD: [{:.3f} -- {:.3f}]".format(
    cvmeans[idx], cvmeans[idx] - 2*cvstds[idx], cvmeans[idx] + 2*cvstds[idx]))
```

```
Accuracy: Mean=0.795, +/- 2 SD: [0.789 -- 0.801]
```

In [11]:

```
#Evaluate performance on Test Set
best_cv_depth = 7
fitted_tree = DecisionTreeClassifier(max_depth=best_cv_depth).fit(X_train, y_tra
in)
best_cv_tree_train_score = fitted_tree.score(X_train, y_train)
best_cv_tree_test_score = fitted_tree.score(X_test, y_test)
print(f"The tree of depth {best_cv_depth} achieved an Accuracy of {best_cv_tree_
test_score:.3f} on the test set.")
```

```
The tree of depth 7 achieved an Accuracy of 0.794 on the test set.
```

Base Random Forest without NLP features

In [12]:

```
#Fit a Random Forest model
fitted_rf = RandomForestClassifier(n_estimators=7, max_depth=7).fit(X_train,y_tr
ain)
random_forest_train_score = fitted_rf.score(X_train, y_train)
random_forest_test_score = fitted_rf.score(X_test, y_test)
print(f"The Random Forest scored {random_forest_train_score:.3f} on the training
set.")
print(f"The Random Forest scored {random_forest_test_score:.3f} on the test set.
")
```

```
The Random Forest scored 0.796 on the training set.
The Random Forest scored 0.794 on the test set.
```

Decision Tree with NLP features

In [13]:

```
all_tweets_df = all_tweets[['retweet_count', 'favorite_count', 'num_hashtags', '
num_urls', 'num_mentions',
                                'user_type', 'sentiment_negative', 'sentiment_ne
utral', 'sentiment_positive',
                                'token_count', 'url_token_ratio', 'ratio_neg',
                                'ant', 'fear', 'joy', 'trust','jaccard']]
train_base_tweets_df, test_base_tweets_df = train_test_split(all_tweets_df, test
_size=0.33, random_state=42, stratify=all_tweets_df['user_type'])
display(train_base_tweets_df.head(2))
print('train tweets shape:',train_base_tweets_df.shape)
print('test tweets shape:',test_base_tweets_df.shape)
```

	retweet_count	favorite_count	num_hashtags	num_urls	num_mentions	user_type	sen
54933	0	0	0	1	0	0	
80644	0	0	0	1	0	0	

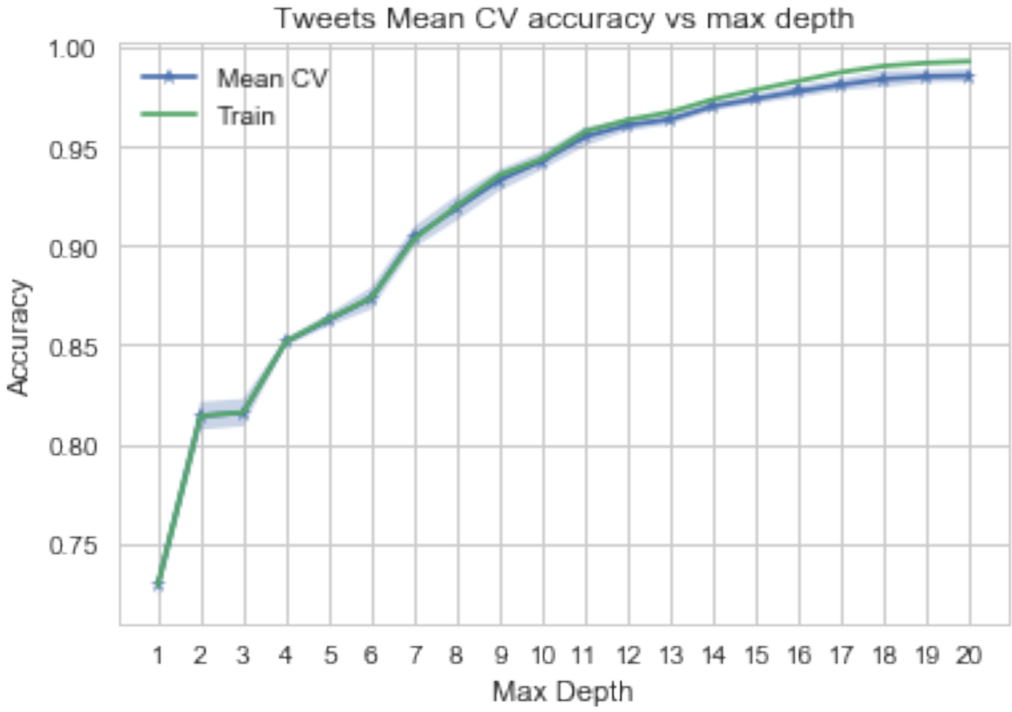
```
train tweets shape: (80574, 17)
test tweets shape: (39686, 17)
```

In [14]:

```
#Perform 5-fold cross validation and store results
X_train, y_train = train_base_tweets_df.drop('user_type',axis=1), train_base_twe
ets_df['user_type']
X_test, y_test = test_base_tweets_df.drop('user_type',axis=1), test_base_tweets_
df['user_type']

depths, train_scores, cvmeans, cvstds, cv_scores = [], [], [], [], []
for depth in range(1,21):
    depths.append(depth)
    dt = DecisionTreeClassifier(max_depth=depth)
    train_scores.append(dt.fit(X_train, y_train).score(X_train, y_train))
    scores = cross_val_score(estimator=dt, X=X_train, y=y_train, cv=5)
    cvmeans.append(scores.mean())
    cvstds.append(scores.std())

#Alter data structure for using internal numpy functions
cvmeans = np.array(cvmeans)
cvstds = np.array(cvstds)
#Plot Means and Shade the +-2 SD Interval
plt.plot(depths, cvmeans, '*-', label="Mean CV")
plt.fill_between(depths, cvmeans - 2*cvstds, cvmeans + 2*cvstds, alpha=0.3)
ylim = plt.ylim()
plt.plot(depths, train_scores, '-+', label="Train")
plt.ylim(ylim)
plt.legend()
plt.ylabel("Accuracy")
plt.xlabel("Max Depth")
plt.title('Tweets Mean CV accuracy vs max depth')
plt.xticks(depths)
plt.show()
```



In [15]:

```
#Choosing the best depth
idx = depths.index(12)
print("Accuracy: Mean={:.3f}, +/- 2 SD: [{:.3f} -- {:.3f}]".format(
    cvmeans[idx], cvmeans[idx] - 2*cvstds[idx], cvmeans[idx] + 2*cvstds[idx]))
```

```
Accuracy: Mean=0.961, +/- 2 SD: [0.958 -- 0.964]
```

In [16]:

```
#Fit a Random Forest model
fitted_rf = RandomForestClassifier(n_estimators=10, max_depth=12).fit(X_train,y_
train)
random_forest_train_score = fitted_rf.score(X_train, y_train)
random_forest_test_score = fitted_rf.score(X_test, y_test)
print(f"The Random Forest scored {random_forest_train_score:.3f} on the training
set.")
print(f"The Random Forest scored {random_forest_test_score:.3f} on the test set.
")
```

```
The Random Forest scored 0.933 on the training set.
The Random Forest scored 0.927 on the test set.
```

In [ ]:

```

```

In [ ]:

```

```