MET Bhujbal Knowledge City,Nashik

DATA MINING AND WAREHOUSING MINI-PROJECT REPORT

SUBMITTED BY

Roll No.	Name
24	**Rohit Sonawane**
23	**Charudutta Sawant**
22	**Vishal Wanve**

Under the guidance of

Prof. Vishal Patil

DEPARTMENT OF COMPUTER ENGINEERING

Academic Year 2021-22

Contents

Problem Statement

Consider a labelled dataset belonging to an application domain. Apply suitable data pre-processing steps such as handling of null values, data reduction, discretization. For prediction of class labels of given data instances, build classifier models using different techniques (minimum 3), analyse the confusion matrix and compare these models. Also apply cross validation while preparing the training and testing datasets.

Abstract

Classification is a form of data analysis that extracts models describing important data classes. Such models, called classifiers, predict categorical (discrete, unordered) class labels. For example, we can build a classification model to categorize bank loan applications as either safe or risky. Such analysis can help provide us with a better understanding of the data at large. In this project we use multiple classification models to analyse the human sentiment of the tweets made during the time of COVID. Apply suitable data pre-processing steps. We then compare performance of classification models to find which one is the best.

Introduction

We have accessed the data from kaggle The tweets have been pulled from Twitter and manual tagging has been

Columns:
1) Location : from where person has tweeted
2) Tweet At day on which tweet is tweeted
3) Original Tweet: Actual text content of the tweet
5) Sentiment: Human sentiment behind the tweet i.e positive, negative

Code

1.File pre processing and cleaning

```
train=pd.read_csv("Corona_NLP_train.csv",encoding='latin1')
test=pd.read_csv("Corona_NLP_test.csv",encoding='latin1')
```

```
train.head()
```

```
train.dtypes
```

```
UserName          int64
ScreenName        int64
Location         object
TweetAt          object
OriginalTweet    object
Sentiment        object
dtype: object
```

```
train['text']=train.OriginalTweet.astype(str)
test['text']=test.OriginalTweet.astype(str)
```

```
train['text'] = train.OriginalTweet
train["text"] = train["text"].astype(str)

test['text'] = test.OriginalTweet
test["text"] = test["text"].astype(str)

# Data has 5 classes converting them to 3

def classes_def(x):
    if x ==  "Extremely Positive":
        return "2"
    elif x == "Extremely Negative":
        return "0"
    elif x == "Negative":
        return "0"
    elif x ==  "Positive":
        return "2"
    else:
        return "1"


train['label']=train['Sentiment'].apply(lambda x:classes_def(x))
test['label']=test['Sentiment'].apply(lambda x:classes_def(x))
```

```
train.label.isna().sum()
```

```
0
```

```
train.label.value_counts(normalize= True)
```

```
2    0.438467
0    0.374128
1    0.187404
Name: label, dtype: float64
```

Text Cleaning

```
#Removing Urls and HTML links
def remove_urls(text):
    url_remove = re.compile(r'https?://\S+|www\.\S+')
    return url_remove.sub(r'', text)
train['text_new']=train['text'].apply(lambda x:remove_urls(x))
test['text_new']=test['text'].apply(lambda x:remove_urls(x))

def remove_html(text):
    html=re.compile(r'<.*?>')
    return html.sub(r'',text)
train['text']=train['text_new'].apply(lambda x:remove_html(x))
test['text']=test['text_new'].apply(lambda x:remove_html(x))

# Lower casing
def lower(text):
    low_text= text.lower()
    return low_text
train['text_new']=train['text'].apply(lambda x:lower(x))
test['text_new']=test['text'].apply(lambda x:lower(x))


# Number removal
def remove_num(text):
    remove= re.sub(r'\d+', '', text)
    return remove
train['text']=train['text_new'].apply(lambda x:remove_num(x))
test['text']=test['text_new'].apply(lambda x:remove_num(x))
```

Fig :File pre processing and text cleaning

2. TF-IDF

Transforming text into vectors

TF-IDF

```
tfidf = TfidfVectorizer(sublinear_tf=True, min_df=5,
                        stop_words='english')

#Transforming each text into a vector
features = tfidf.fit_transform(train.text).toarray()

labels = train.label

print("Each of the %d tweets is represented by %d features (TF-IDF score of unigrams and bigrams)" %(features.shape))
```

```
Each of the 41157 tweets is represented by 10500 features (TF-IDF score of unigrams and bigrams)
```

3. Models

Models

```
models = [
    RandomForestClassifier(n_estimators=100, max_depth=5, random_state=0),
    LinearSVC(),
    MultinomialNB(),
]
```

Fig: Models

4. Cross validation

Cross Validation

```
# 5 Cross-validation
CV = 5
cv_df = pd.DataFrame(index=range(CV * len(models)))

entries = []
for model in models:
  model_name = model.__class__.__name__
  accuracies = cross_val_score(model, features, labels, scoring='accuracy', cv=CV)
  for fold_idx, accuracy in enumerate(accuracies):
    entries.append((model_name, fold_idx, accuracy))

cv_df = pd.DataFrame(entries, columns=['model_name', 'fold_idx', 'accuracy'])
```

```
mean_accuracy = cv_df.groupby('model_name').accuracy.mean()
std_accuracy = cv_df.groupby('model_name').accuracy.std()

acc = pd.concat([mean_accuracy, std_accuracy], axis= 1,
          ignore_index=True)
acc.columns = ['Mean Accuracy', 'Standard deviation']
acc
```

model_name	Mean Accuracy	Standard deviation
LinearSVC	0.792162	0.003353
MultinomialNB	0.660276	0.003075
RandomForestClassifier	0.460067	0.006552

Linear SVM has the highest accuracy of 78%, followed by Naive Bayes

Fig. Cross Validation

5. Comparison

```
plt.figure(figsize=(8,5))
sns.boxplot(x='model_name',y='accuracy',data=cv_df,color='purple',showmeans=True)
plt.title("MEAN ACCURACY (cv = 5)\n", size=14);
```

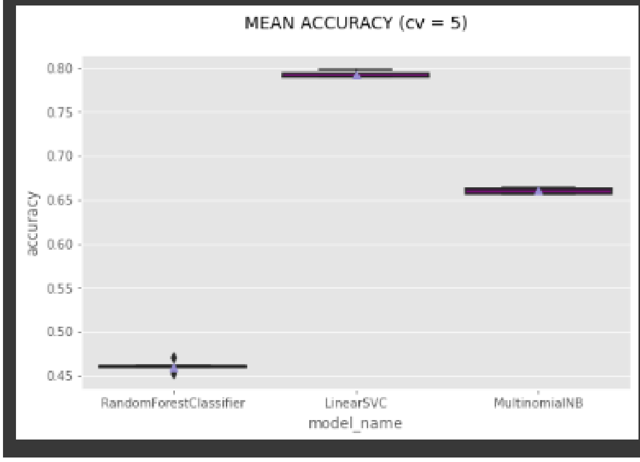

Fig. Model Comparison

6. Classification

```
X_train, X_test, y_train, y_test,indices_train,indices_test = train_test_split(features,
                                                               labels,
                                                               train.index, test_size=0.10,
                                                               random_state=1)
model =   LinearSVC()
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
```

```
# Classification report

from sklearn.model_selection import cross_val_score
from sklearn.metrics import confusion_matrix
from sklearn import metrics

print('\t\t\t\tCLASSIFICATIION METRICS\n')
print(metrics.classification_report(y_test, y_pred,
                                    target_names= train['label'].unique()))
```

```
                                CLASSIFICATIION METRICS

              precision    recall  f1-score   support

           1       0.81      0.80      0.80      1554
           2       0.70      0.68      0.69       760
           0       0.82      0.84      0.83      1802

    accuracy                           0.80      4116
   macro avg       0.78      0.77      0.78      4116
weighted avg       0.79      0.80      0.79      4116
```

Fig. Classification Report

We see low F1 scores for Positive (2) tweets as compared to Negative (0) and Neutral (1)

7. Confusion Matrix

```
conf_mat = confusion_matrix(y_test, y_pred)
fig, ax = plt.subplots(figsize=(8,8))
sns.heatmap(conf_mat, annot=True, cmap="Accent_r", fmt='d',
            xticklabels=sentiment_id_df.label.values,
            yticklabels=sentiment_id_df.label.values)
train = train.drop(columns=['sentiment'])
plt.ylabel('Actual')
plt.xlabel('Predicted')
plt.title("CONFUSION MATRIX - LinearSVC\n", size=16);
```

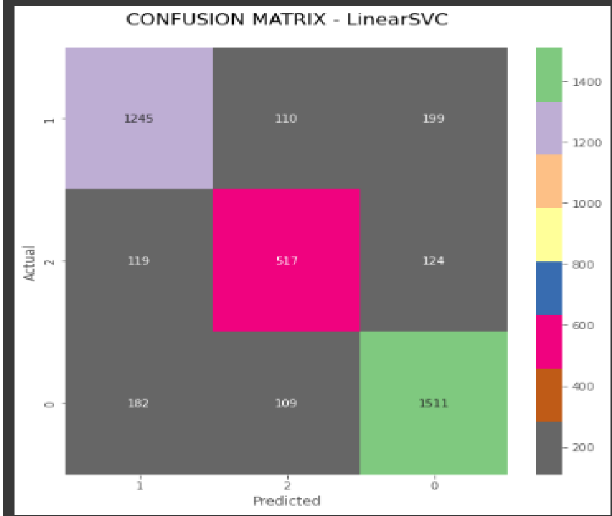

Fig. Confusion Matrix

Result

The accuracy for Linear SVM is around 78%. while that of other models is lesser.

Conclusion

We have analysed COVID tweets dataset and performed data pre-processing steps. We have experimented multiple classification models and found out the best performer among them

References

[1] https://www.kaggle.com/datatattle/battle-of-ml-classification-models/data

[2] Introduction to Machine Learining