TEXT MINING AND SENTIMENT ANALYSIS PROJECT

Şirag

PREPARED BY AZIZAGHA ABASZADA

SUMMARY INFORMATION

In this project, a topic was chosen and the texts written on Twitter about that topic were taken. This text was sampled from the data and divided into certain categories. Subsequently, these sample data were taught to artificial intelligence algorithms and all data were divided into certain categories with these algorithms. Finally, sentiment analysis was performed on the categorized data, and it was revealed that people said negative or positive information for each category.

Subject Heading: Online Education

Data Source: Twitter

Data Size: 6391 tweet (unedited data)

About Subject:

The preparation of this project was made during the Covid-19 pandemic period. And one of the topics on the agenda of the term was online education. And in this project there is information about **Online education** that people share on Twitter.
Note: These data are taken from tweets in Turkish.

Note: Since this project has to be done in Turkish, Turkish data were used.

Python Code:

https://gitlab.com/Aziz9900/Sentiment-Analysis/-/blob/main/Predictive%20Analysis%20and%20Sentiment%20Analysis.ipynb

1. step:

Data set is about the word "Uzaktan Eğitim" (it is mean Online Education) was pulled from the Twitter developer account. After these data were added to excel named "Kirli_veri" (it is mean unedited data).

2. step:

Then by doing operations on the unedited data using python. Clean data has been extracted. In the data cleaning process, unnecessary words and repetitive data in the data set were removed.

3. step:

After the cleaning process, the data was saved in an excel named "temiz_veri" (it is mean clean data). A sample was taken from this clean data data and 4 different categorical data sets were created with 200 data in each category. This dataset was used to train the machine learning algorithms.

4. step:

Model Training

1. **Import Python libraries and Data set.**

```
import pandas as pd
import os
from snowballstemmer import TurkishStemmer
turkStem = TurkishStemmer()
# Adding libraries
```

2. **Import category data set.**

```
ogrenme_df = pd.read_excel("kategori.xlsx")
# Adding the dataset for pre-made model training
```

3. 100 data were drawn for each category from the data set and a category data set was created. These 4 categories are live class, education, exam, technology. (in Turkish canli ders, eğitim, sınav, teknoloji).

```
ogrenme_df.head()
```

	category	text
0	sınav	sınava gelmiyorum diyip gitmeyenler yarın öbür...
1	sınav	net sınavlar iptal olsun zaten kimsenin anladı...
2	sınav	ben sınavların iptal olacağına olan inancımı k...
3	sınav	online verdiğin derslerin sınavını online yap
4	sınav	bahçeşehir kolejleri arasında yapılan agis sın...

4. Dividing texts into roots and Categorizing dataset as numbers

```
ogrenme_df['text'] = ogrenme_df['text'].apply(lambda x: " ".join([turkStem.stemWord(i) for i in x.split()]))
# Dividing texts into roots
```

```
ogrenme_df['numara'] = pd.factorize(ogrenme_df.category)[0]
#Categorizing dataset as numbers
```

```
ogrenme_df.groupby(["category", "numara"]).size()
```

```
category    numara
canlı ders  1         100
eğitim      2         100
sınav       0         100
teknoloji   3         100
dtype: int64
```

5. **Splitting the dataset into testing and learning**

```
from sklearn.model_selection import train_test_split

x_train, x_test, y_train, y_test = train_test_split(ogrenme_df["text"], ogrenme_df["numara"], test_size = 0.2, random_state = 4)
#Splitting the dataset into testing and learning
```

```
x_train.shape
```

```
(320,)
```

```
x_test.shape
```

```
(80,)
```

6. **Text data is translated into numeric. and thus each text is assigned a numerical value.**

```
vectorizer = TfidfVectorizer()
```

```
train_vectors = vectorizer.fit_transform(x_train)
test_vectors = vectorizer.transform(x_test)
print(train_vectors.shape, test_vectors.shape)
#text data is translated into numeric. and thus each text is assigned a numerical value.
```

```
(320, 1327) (80, 1327)
```

7. Preparation of necessary libraries for processing prediction models

```
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.naive_bayes import MultinomialNB
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.model_selection import cross_val_score
from sklearn import svm
import seaborn as sns

#Preparation of necessary libraries for processing prediction models
```

8. Testing data in different algorithms
(Data sets were tested with 6 different machine learning algorithms.)

```
LogicReg = LogisticRegression()
LogicReg.fit(train_vectors, y_train)
prediction = LogicReg.predict(test_vectors)
print(accuracy_score(y_test, prediction))
model_score.loc[0] =["Logistic Regression", accuracy_score(y_test, prediction)]
```

```
clf = MultinomialNB()
clf.fit(train_vectors, y_train)
prediction = clf.predict(test_vectors)
print(accuracy_score(y_test, prediction))
model_score.loc[1] =["Naive Bayes", accuracy_score(y_test, prediction)]
```

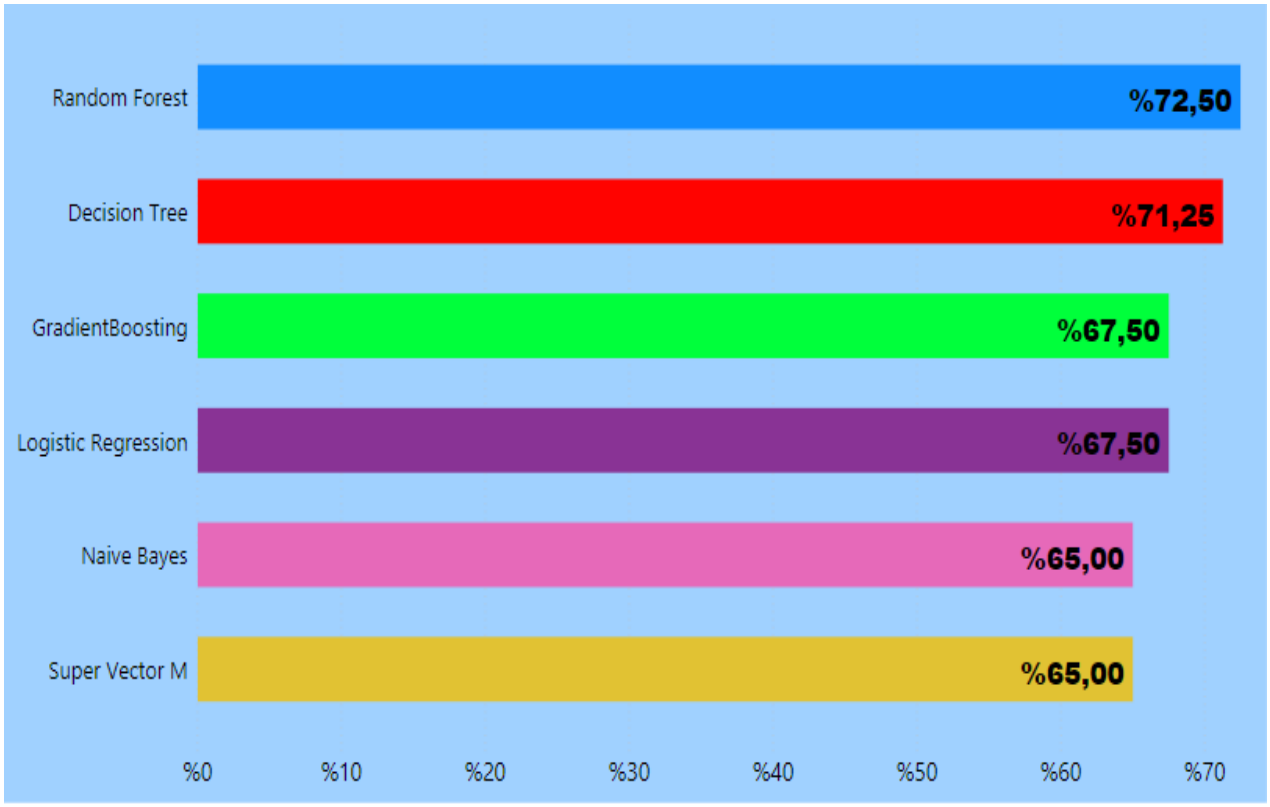
```
dTmodel = DecisionTreeClassifier()
dTmodel.fit(train_vectors, y_train)
prediction = dTmodel.predict(test_vectors)
print(accuracy_score(y_test, prediction))
model_score.loc[2] =["Decision Tree", accuracy_score(y_test, prediction)]
```

```
rForest = RandomForestClassifier()
rForest.fit(train_vectors,y_train)
prediction=rForest.predict(test_vectors)
print(accuracy_score(y_test, prediction))
model_score.loc[3] =["Random Forest ", accuracy_score(y_test, prediction)]
```

```
grBoosting = GradientBoostingClassifier()
grBoosting.fit(train_vectors, y_train)
prediction = grBoosting.predict(test_vectors)
print(accuracy_score(y_test, prediction))
model_score.loc[4] =["GradientBoosting", accuracy_score(y_test, prediction)]
```

```
super_vectorM = svm.SVC()
super_vectorM.fit(train_vectors, y_train)
prediction = super_vectorM.predict(test_vectors)
print(accuracy_score(y_test, prediction))
model_score.loc[5] =["Super Vector M", accuracy_score(y_test, prediction)]
```

9. Success score on 6 different machine learning algorithm datasets



5. Step:

Predict new data with the Selected Models

The two most successful algorithms (with the highest prediction rate) were chosen to make predictions on the real data set. These are **Random Forest** and **Decision Tree** algorithms. These algorithms will be tested on real data.

1. **Text data is translated into numeric. and thus each text is assigned a numerical value.**

```
test_vectors_ = vectorizer.transform(df["text"].astype('U').values)
print(test_vectors_.shape)
#text data is translated into numeric. and thus each text is assigned a numerical value.
```

```
(1636, 1327)
```

2. **Making predictions using the Decision Tree algorithm**

```
predicted = dTmodel.predict(test_vectors_)
tahmin = pd.DataFrame(predicted)
tahmin.rename(columns = {0:'tahmin'}, inplace = True)
df["tahmin_dTmodel"] = tahmin
# Doing data segmentation using the Decision Tree model
```

```
df.loc[df['tahmin_dTmodel'] == 0, ['dTmodel_tahmin']] = 'Sınav'
df.loc[df['tahmin_dTmodel'] == 1, ['dTmodel_tahmin']] = 'canlı ders'
df.loc[df['tahmin_dTmodel'] == 2, ['dTmodel_tahmin']] = 'eğitim'
df.loc[df['tahmin_dTmodel'] == 3, ['dTmodel_tahmin']] = 'teknoloji'

#I defined categories to categorize the data.
# 0 - The Exam
# 1 - Online Class
# 2 - Education
# 3 - Technology
```

3. Making predictions using the Random Forest algorithm

```
predicted = rForest.predict(test_vectors_)
tahmin = pd.DataFrame(predicted)
tahmin.rename(columns = {0:'tahmin'}, inplace = True)
df["tahmin_rForest"] = tahmin
# Doing data segmentation using the Random Forest model
```

```
df.loc[df['tahmin_rForest'] == 0, ['rForest_tahmin']] = 'Sınav'
df.loc[df['tahmin_rForest'] == 1, ['rForest_tahmin']] = 'canlı ders'
df.loc[df['tahmin_rForest'] == 2, ['rForest_tahmin']] = 'eğitim'
df.loc[df['tahmin_rForest'] == 3, ['rForest_tahmin']] = 'teknoloji'
```

4. Results of predictions

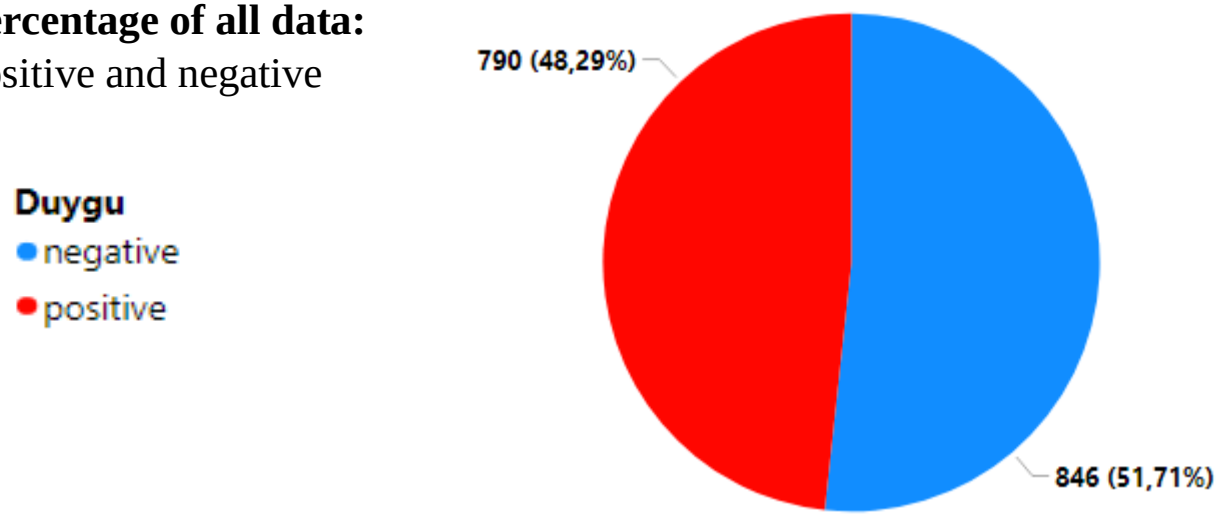
```
df.groupby("dTmodel_tahmin").size()
# The result of the Decision Trees model
# Exam - 92
# Online class - 192
# Education - 1247
# Technology - 185
```

```
df.groupby("rForest_tahmin").size()
# The result of the Random Forest model
# Exam - 83
# Online class - 138
# Education - 1333
# Technology - 82
```

6. Step: Sentiment Analysis (positive or negative estimation of data by categories)

After categorizing the data using algorithms, sentiment analysis was performed according to these categories. Turkish bert library was used for sentiment analysis.

Percentage of all data:
Positive and negative



Number of data in category:
Positive and negative