האוניברסיטה העברית בירושלים
THE HEBREW UNIVERSITY OF JERUSALEM

Final Project

Text Mining Course

Prof. Ronen Feldman

Ziv Parchi 204589949

Dor Simai 204038012

Table of Contents

1. Introduction

1.1. Introduction

The FOMC (Federal Open Market Committee) holds eight regularly scheduled meetings per year. At these meetings, the Committee reviews economic and financial conditions, determines the appropriate stance of monetary policy, and assesses the risks to its long-run goals of price stability and sustainable economic growth. The influence of the Federal Reserve's decisions has been the subject of numerous studies, and still, little is known about the real-time response of U.S. market.

After each FOMC meeting there are a few outputs that are released to the press and public, and for our project, we chased to focus on the Minutes in order to predict the market direction, as we believe they contain all the important content of the meeting under a similar structure which could help us while using Text Mining technics. The FOMC minutes provide a detailed summary of the FOMC discussions and inform the public and Congress about the full range of policy-makers views and debates about monetary policy issues.

We worked on the following project as part of the Text Mining course in which we learned about different Text Mining approaches. Throughout the work on this project, we used materials and examples that were taught in class and also deepened our learning from online materials.

1.2. Motivation

The topic and research question of this project was given to us by the lecturer Prof. Ronen Feldman, but it is important for us to note that we have a lot of passion for the two main fields on which the project focuses - Text Mining & Capital Market. We think that there is enormous potential in the combination of them and that there is room for much more academic and business researches from this kind.

1.3. Research questions

The question that will lead this research is as follows: *Is it possible to predict the market price by text mining of the Fed Minutes?*

The target variable in our project will be the market price which will be expressed by the S&P 500 index.

2. Organizing the data

2.1. Scraping:

Scraping is a process of collating a collection of webpages by starting with an initial list of URLs (or links) and systematically processing each page to extract the content and additional links.

In this step, we used the www.federalreserve.gov website to allocate the relevant HTML text. Using the BeautifulSoup package, we were able to read the data from the website and make sure we are getting the FED Minutes' text & release dates of every FOMC meeting.

2.2. Parsing and cleaning the data:

As part of this step, we created useful functions for parsing and cleaning the Minutes' data and created a data frame that is consisted of the release dates of the Minutes and their content.

	date	content
0	February 16, 2022	The agenda for this meeting reported that advi...
1	February 17, 2021	The agenda for this meeting reported that advi...
2	April 07, 2021	By unanimous vote, the Committee approved a fi...
3	May 19, 2021	The System Open Market Account (SOMA) manager ...
4	July 07, 2021	Participants resumed their discussion from the...

2.3. Preprocessing:

As part of this step, we cleaned the data from the following cases:

- Uppercases
- Stopwords
- Unicode - encoding the text to ASCII format
- Unicode - decoding the text
- Extra whitespace
- Mentions
- Market tickers
- URLs
- Hashtags
- Punctuation

```
def clean_text(text):
    # Lowercasing the text
    text = text.lower()
    # Removing stopwords
    stop_words = set(stopwords.words("english"))
    text = " ".join([word for word in text.split() if word not in stop_words])
    # removing unicode - encoding the text to ASCII format
    text = text.encode(encoding="ascii", errors="ignore")
    # removing unicode - decoding the text
    text = text.decode()
    # cleaning the text to remove extra whitespace
    text = " ".join([word for word in text.split()])
    # removing mentions
    text = re.sub("@\S+", "", text)
    # remove market tickers
    text = re.sub("\$", "", text)
    # remove urls
    text = re.sub("https?:\/\/.*[\r\n]*", "", text)
    # removing hashtags
    text = re.sub("#", "", text)
    # remove punctuation
    punct = set(string.punctuation)
    text = "".join([ch for ch in text if ch not in punct])

    return text
```

Finally, we got an additional column of a cleaned content:

	date	content	cleaned_content
0	February 16, 2022	The agenda for this meeting reported that advi...	agenda meeting reported advices election follo...
1	February 17, 2021	The agenda for this meeting reported that advi...	agenda meeting reported advices election follo...
2	April 07, 2021	By unanimous vote, the Committee approved a fi...	unanimous vote committee approved final rule r...
3	May 19, 2021	The System Open Market Account (SOMA) manager ...	system open market account soma manager first ...
4	July 07, 2021	Participants resumed their discussion from the...	participants resumed discussion april 2021 fom...

2.4. Adding the target variable:

Since we wanted to use the market direction and check the S&P 500 index status change (and in other words - if the market went “up” or “down”) after the Fed Minutes’ release, we decided to calculate the average of the index in the week before the FOMC meeting and in the week ahead. Afterward, we checked if the index went up or down following the meeting. In this step, we used Yahoo Finance API.

	date	content	cleaned_content	avg_index_week_before	avg_index_week_ahead	weekly_index_diff	weekly_index_status
0	2017-02-22	Participants considered a revised proposal fro...	participants considered revised proposal subco...	235.305000	236.608002	1.303001	1
1	2017-04-05	The manager of the System Open Market Account ...	manager system open market account soma report...	235.675998	235.163998	-0.512000	0
2	2017-05-24	The manager of the System Open Market Account ...	manager system open market account soma report...	238.094003	241.395000	3.300997	1
3	2017-07-05	By unanimous vote, the Committee selected Mark...	unanimous vote committee selected mark lj wrig...	242.212505	241.998001	-0.214504	0
4	2017-08-16	By unanimous vote, the Committee selected Mark...	unanimous vote committee selected mark e van d...	245.635995	244.216000	-1.419995	0

* It is important to note that during our research we also tried to use the day before the FOMC meeting and the day after the Minutes’ release but got lower results in terms of accuracy in all of our models.

3. Sentiment analysis

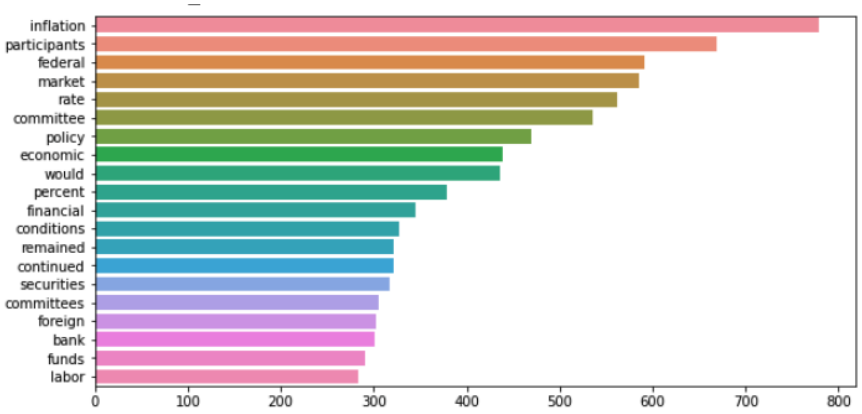
As part of the sentiment analysis, we used a list of positive words and a list of negative words from “Loughran and McDonald Sentiment Word Lists” and detected their amount in each Minute’s text. Finally, we calculated the Sentiment Score by normalizing them according to the number of the total words in each text.

```
temp = [tone_count_with_negation_check(lmdict,x) for x in df_fed.cleaned_content]
temp = pd.DataFrame(temp)

df_fed['wordcount'] = temp.iloc[:,0].values
df_fed['NPositiveWords'] = temp.iloc[:,1].values
df_fed['NNegativeWords'] = temp.iloc[:,2].values

df_fed['sentiment'] = (df_fed['NPositiveWords'] - df_fed['NNegativeWords']) / df_fed['wordcount'] * 100

df_fed['Poswords'] = temp.iloc[:,3].values
df_fed['Negwords'] = temp.iloc[:,4].values
```

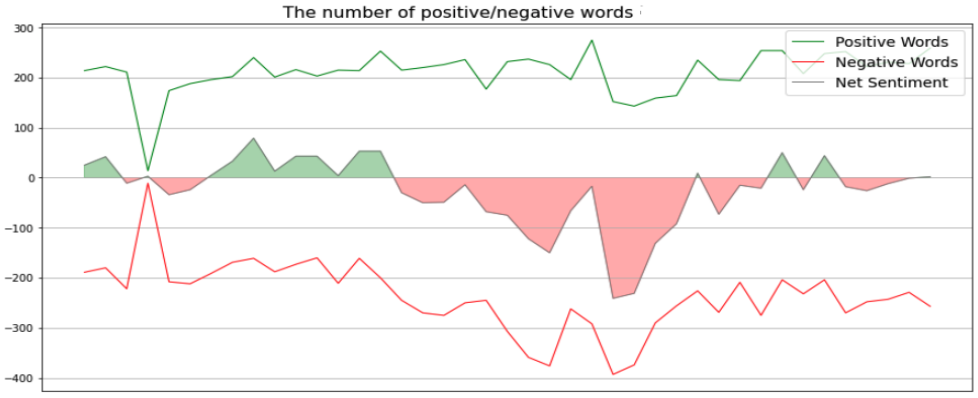
After this process, we got the following data frame:

daily_index_status	wordcount	NPositiveWords	NNegativeWords	sentiment	Poswords	Negwords
0	4553	214	189	0.549089	[benefiting, stronger, growth, smoothly, highe...	[uncertainty, uncertainty, uncertainty, volati...
0	5184	222	180	0.810185	[increased, rose, progress, high, higher, adva...	[uncertainty, volatility, declined, late, late...
1	4982	211	222	-0.220795	[increase, increase, smoothly, certain, certai...	[declined, fell, weaker, volatility, declined,...
0	585	14	11	0.512821	[effective, rise, high, increase, rising, rise...	[declined, easing, volatility, low, lower, fal...

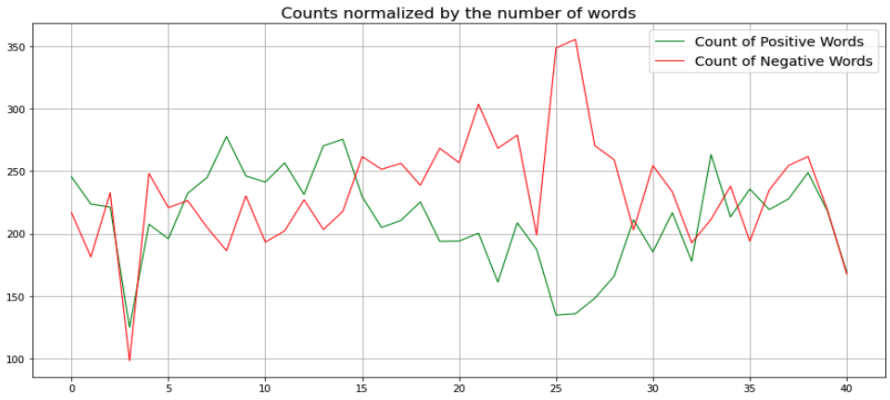
When we plotted the number of words that were most common, we noticed (as accepted), that many of them were belong to financial and economic terms.



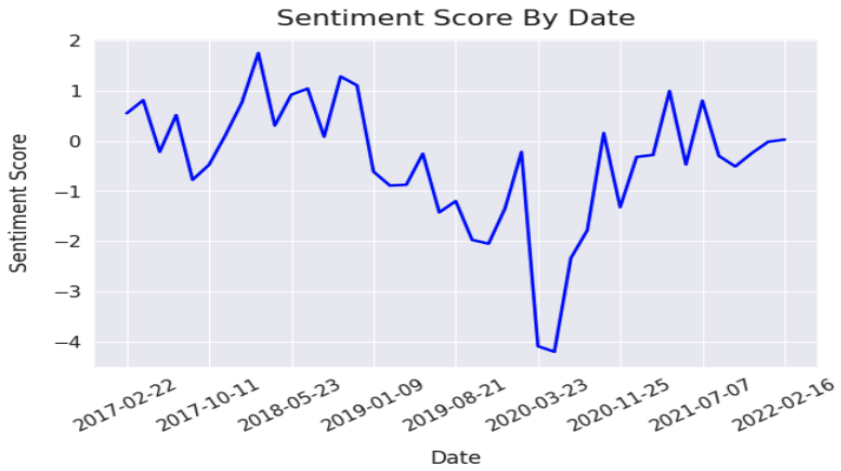
When we presented the number of positive & negative words on a plot, we were able to see that throughout the years, the positive word count and negative count were pretty correlated, and the average is on the negative side.



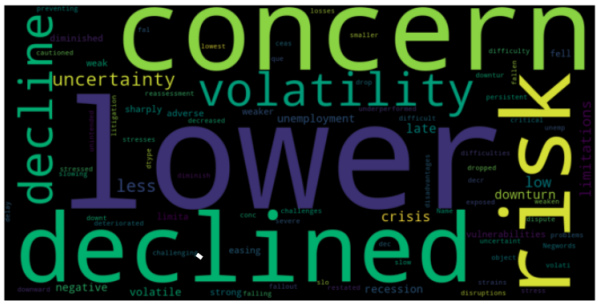
When we plotted the counts that were normalized by the number of words, we noticed that there is a correlation in most of the time. It is important to note that in both plots the COVID-19 affect can be seen since March-2020 - in the time were the market went down.



When we visualized the sentiment score by the date, we noticed a sharp decrease in March-2020 that reflects the world & market situation when the Covid-19 started to affect the markets.



Using Word Cloud, we were able to visualize the most common positive words (in white) and the most common negative words (in black). From this visualization, we can see that WordCloud is able to detect positive and negative words in our text.



4. Preparing the data for the models:

4.1. TF-IDF:

TF-IDF is a word-document mapping with a normalization. It ignores the order of words and provides a matrix where n is the number of words in the vocabulary and m is the number of documents. Using the DataFrameMapper which is part of the *sklearn_pandas* package, we were able to create a tf-idf data frame that is composed of 41 rows and 7,669 columns.

```
mapper = DataFrameMapper([
     ('cleaned_content', TfidfVectorizer()),
     ('weekly_index_status', None)], df_out=False)

mapper_fit = mapper.fit(df_fed)
final_df = mapper.transform(df_fed)

print(final_df.shape)
```

```
(41, 7669)
```

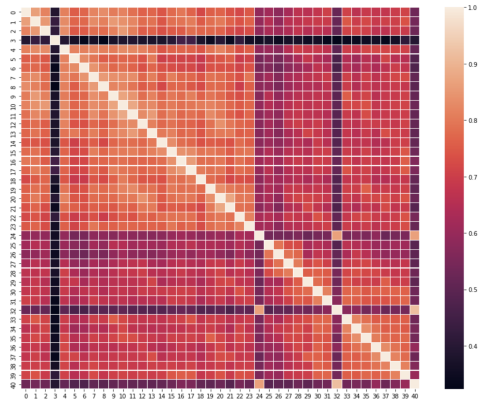
```
df_tfidf = pd.DataFrame(final_df)
df_tfidf = df_tfidf.reset_index(drop=True)

#Creating y and x seperately so we could normalize and split to train and test
Y = df_tfidf.iloc[:,-1]
X = df_tfidf.iloc[:,:-1]

X = preprocessing.normalize(X)
X = pd.DataFrame(X)
```

```
df_tfidf.head()
```

	0	1	2	3	4	5	6	7	8	9	...	7659	7660	7661	7662	7663	7664	7665	7666	7667	7668
0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.000000	0.000000	0.01504	...	0.0	0.008377	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0
1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.010979	0.009651	0.00000	...	0.0	0.007380	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	0.000000	0.000000	0.00000	...	0.0	0.008058	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0
3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.000000	0.000000	0.00000	...	0.0	0.000000	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	0.000000	0.000000	0.00000	...	0.0	0.000000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

Using Cosine Similarity, we noticed that at some point in the timeline (Minute #24 = 2020-02-19) the structure of the Fed Minutes has changed and it can be seen by the following plot. We believe that this change is related to the COVID-19 pandemic.

```
cosine_df = cosine_similarity(X)
fig = plt.figure(figsize=(15,12))
ax = sns.heatmap(cosine_df,linewidths=.1)
```

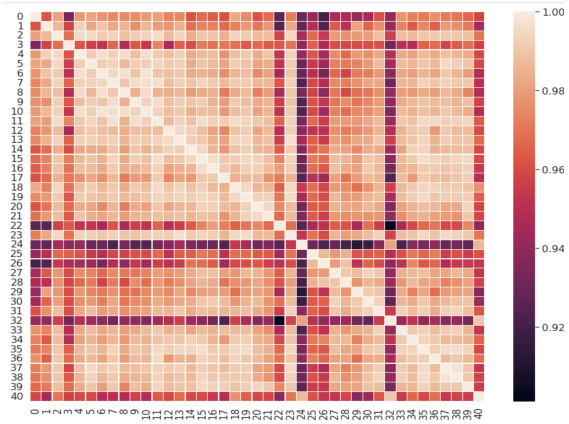


4.2. Doc2vec:

Doc2Vec gives a unique vector for each document based on the words appearing in the particular document. In parallel to tf-idf, we also wanted to examine the Doc2vec since, unlike tf-idf, it takes into account the placement of words in a document.

```
# Tokenization of each document:
tokenized_doc = [word_tokenize(d) for d in df_fed["cleaned_content"]]

# Convert tokenized document into gensim formated tagged data:
tagged_data = [TaggedDocument(d, [i]) for i, d in enumerate(tokenized_doc)]

dv_model = Doc2Vec(tagged_data, vector_size=100, window=5, min_count=2, workers=12, epochs=5)
```

Using Cosine Similarity we can see a higher similarity than we got for tf-idf.



4.3. Splitting the data into Train & Test:

After many tests on different kinds of splits, we decided to split the data for Train & Test by 80/20. Basically, this way we were able to reach the best results in our various models. It is important to note that we are aware of the fact that the dataset we have is not large enough. We believe that if we had a larger dataset, we could have reached a higher accuracy and used additional models that require larger data.

```
x_train, x_test, y_train, y_test = train_test_split(X, Y, test_size=0.2, random_state=42)
print('Training set size: {0:d}\nTest set size: {1:d}'.format(len(x_train),len(x_test)))
```

```
Training set size: 32
Test set size: 9
```

5. Models Exploration

As part of this step, we wanted to explore different kinds of models in order to find the most suitable one to answer the business question. As part of our models' evaluation, we used the following measurements to choose the best model:

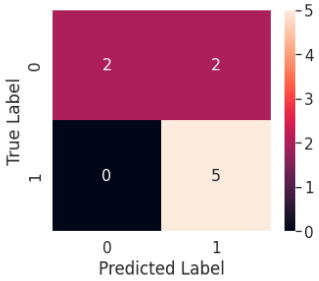
- **Accuracy-** the ratio of the total number of correct predictions out of the total number of predictions.
- **Precision** - the ratio between the True Positives and all the positives.
- **Recall** - the ratio between the True Positives and the True Positive + False Negative.
- **F1** - the Harmonic mean of the Precision and Recall.
- **Confusion Matrix** - can help us learn about the count distribution of the types of correct and incorrect predictions (TP, TN, FP, FN).

In each one of the following models, we examine the result for TF-IDF & DOC2VEC to see which is better for each of the models.

5.1. Logistic Regression:

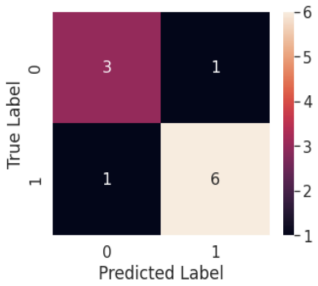
TF-IDF

```
Precision score: 0.714
Recall score: 1.000
F1 score: 0.833
Accuracy: 0.778
Confusion matrix:
[[2 2]
 [0 5]]
```



DOC2VEC

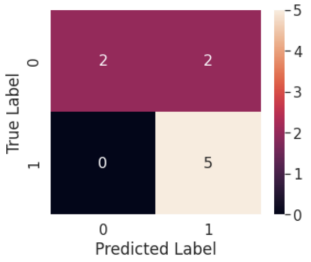
```
Precision score: 0.857
Recall score: 0.857
F1 score: 0.857
Accuracy: 0.818
Confusion matrix:
[[3 1]
 [1 6]]
```



5.2. Decision Tree Classifier:

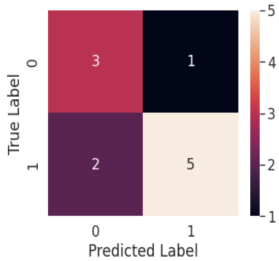
TF-IDF

```
Best Accuracy: 69.17%
Best Params: {'criterion': 'gini', 'max_depth': 1}
best estimator score is 0.78%
Precision score: 0.714
Recall score: 1.000
F1 score: 0.833
Accuracy: 0.778
Confusion matrix:
[[2 2]
 [0 5]]
```



DOC2VEC

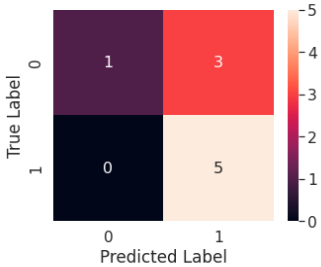
```
Best Accuracy: 73.33%
Best Params: {'criterion': 'gini', 'max_depth': 3}
best estimator score is 0.73%
Precision score: 0.833
Recall score: 0.714
F1 score: 0.769
Accuracy: 0.727
Confusion matrix:
[[3 1]
 [2 5]]
```



5.3. Random Forest:

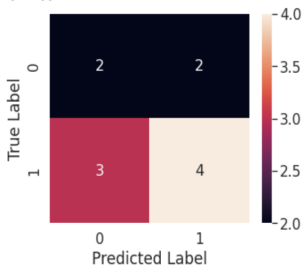
TF-IDF

```
Precision score: 0.625
Recall score: 1.000
F1 score: 0.769
Accuracy: 0.667
Confusion matrix:
[[1 3]
 [0 5]]
```



DOC2VEC

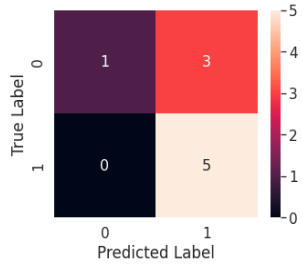
```
Precision score: 0.667
Recall score: 0.571
F1 score: 0.615
Accuracy: 0.545
Confusion matrix:
[[2 2]
 [3 4]]
```



5.4. XGboost:

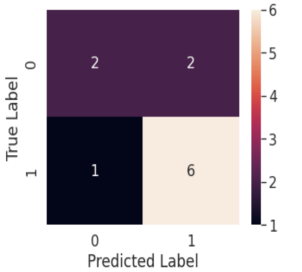
TF-IDF

```
Precision score: 0.625
Recall score: 1.000
F1 score: 0.769
Accuracy: 0.667
Confusion matrix:
[[1 3]
 [0 5]]
```



DOC2VEC

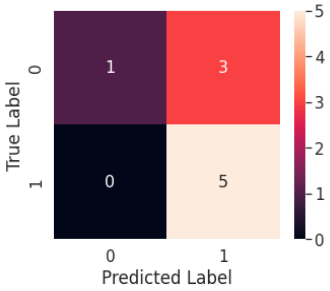
```
Precision score: 0.750
Recall score: 0.857
F1 score: 0.800
Accuracy: 0.727
Confusion matrix:
[[2 2]
 [1 6]]
```



5.4. KNN:

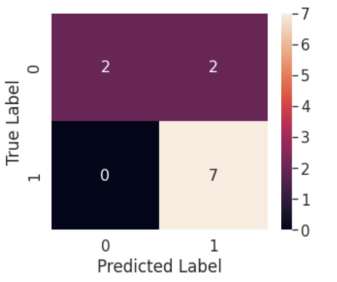
TF-IDF

```
Best Accuracy: 78.57%
Best Params: {'kneighborsclassifier__n_ne
Precision score: 0.625
Recall score: 1.000
F1 score: 0.769
Accuracy: 0.667
Confusion matrix:
[[1 3]
 [0 5]]
```



DOC2VEC

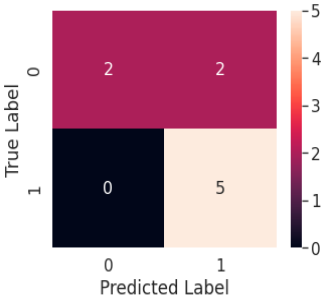
```
Best Accuracy: 76.67%
Best Params: {'kneighborsclassifier__n_ne
Precision score: 0.778
Recall score: 1.000
F1 score: 0.875
Accuracy: 0.818
Confusion matrix:
[[2 2]
 [0 7]]
```



5.5. SGDClassifier:

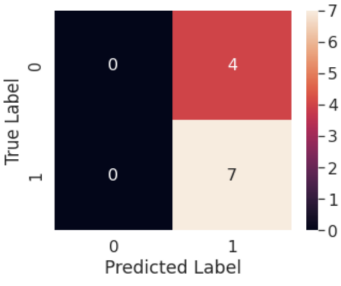
TF-IDF

```
Precision score: 0.714
Recall score: 1.000
F1 score: 0.833
Accuracy: 0.778
Confusion matrix:
[[2 2]
 [0 5]]
```



DOC2VEC

```
Precision score: 0.636
Recall score: 1.000
F1 score: 0.778
Accuracy: 0.636
Confusion matrix:
[[0 4]
 [0 7]]
```



7. Conclusions:

As part of the project, we were requested to answer the question: *Is it possible to predict the market price by text mining of the Fed Minutes?*

In terms of evaluating the various models, we mostly focused on the F1 measurement that combines the Recall and Precision. In addition, we also referred to Precision since the price of the mistake in the case of the research question is very high since we do not want people to use these models and lose their money.

After evaluating all the classification models above, we came to the conclusion that the best model that could answer the research question using our data is the **Logistic regression** in both TF-IDF and DOC2VEC methods, for these main reasons:

- The model reached the highest F1 and Precision scores
- This kind of model is simple to use in terms of efficiency and maintenance

Note that some other classification models we used reach nice results and we believe that with additional data we will be able to evaluate them and our chosen model better.