MILESTONE 3

INDUSTRIAL SECURITY CLEARANCE ADJURATIONS

Under the guidance of

Professor Dr. Alsmadi izzat

By

Srilekhya Janamanchi

Contents

Overall /Research Goal

The major goal of my research project is to analyze and extrapolate data from a substantial dataset related to the determination of industrial security clearances to obtain sufficient characteristics from that dataset to be able to train an analysis model.

Abstract

The research project aims to analyze and extract useful information from the "DOD Clearance Adjudications" dataset to train a machine learning model for industrial security clearance determination. The dataset contains detailed information on security clearance background checks and clearance levels, providing a valuable resource for understanding the security clearance process and identifying potential biases. The project used data preprocessing techniques to extract relevant information from the dataset, including mapping classes with similar meanings, removing stop words and filtering out rare classes. Visualization techniques, such as pie charts and word clouds, were also used to better understand the data. Prior work on the project has focused on background research, the use of machine learning algorithms for classification tasks, and study on legal and policy frameworks. The study emphasizes the value of taking into account all available data throughout the clearance process and the necessity for more reliable data analytics to back clearance judgments. Furthermore, discussed the process of feature selection, including reviewing other entries on Kaggle and creating an extracted digest column. It is noted that non-numerical data cannot be used to calculate correlation coefficients. Data from classification of clearance cases using machine learning showed that decision trees performed better than other models. The results of the train and test models show that the developed model has a 99% accuracy rate, indicating the possibility of deploying the developed machine learning algorithm practically. Additionally, implemented analyzing the dataset sentimentally, improved the model's robustness by handling raw data as input, and identified ways to optimize the model's performance. Lastly, I compared my research to other studies that used the same dataset.

Research Questions

The collection of queries that this project can address include (but are not limited to) the following:

1. Is there any possibility of deploying developed machine learning algorithm practically?
2. Does this developed model can be more robust by handling the raw data as input?
3. Can sentimental analysis be performed on the developed algorithm?

Introduction

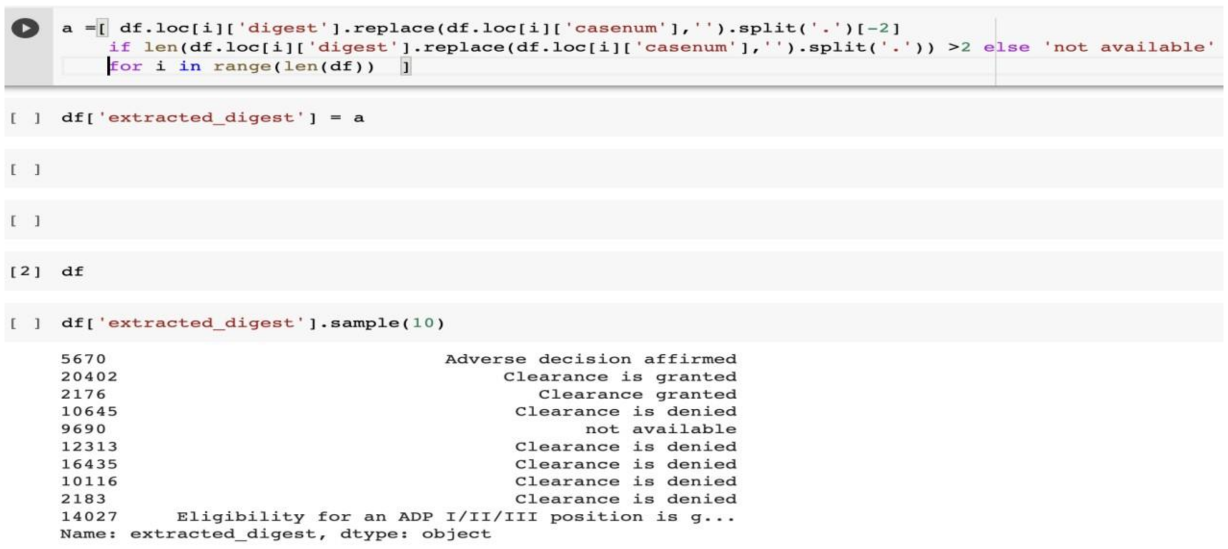
To make the data more reliable to achieve the goal of my project I have modified the given data from the dataset. Which can be accomplished by adding extra new columns and doing some modifications to the code.

Analysis Progress

Using the provided digest column, I have extracted the refined data and placed it in a new column called extracted digest. Once the data has been extracted, a new extracted digest column appears in the table. I have mapped classes with similar meaning to one class**.** This is one of the project's key steps. Where, I have reduced the number of classes. As I map similar classes, the code will become more efficient and exact. In addition, a new column "new classes" is added, where I have dumped all the reduced classes obtained from the previous step using mapping classes. Once mapping is done, I start removing the stop words. This is the process of deriving the root of sentences and words, includes the removal of stop words. Stop words are identified as pointless words and must be eliminated from the project. This is also commonly referred to as data cleaning. This generally covers punctuation, capitalization, and any additional whitespace left over after removing extraneous characters. I have shown the topmost 20 occurring words. To test the visualization in the data set, the top 20 most frequent classes were plotted.

The visualization for the various categories is shown in the pie chart. I have filtered out the classes that occur rarely. The visualization shows that the number of classes has been reduced from nearly 175 to 20. As a result, can easily locate the most frequently used classes. I can see that the most frequently used words are more prominent than the others. This could be used to compare my other processed texts to see how far I have come in cleaning up the words. Characters and words are incomprehensible to machines. So, when dealing with text data, I must represent it numerically so that the machine can understand it. The Count vectorizer method is used to convert text to numerical data by using the extracted digest column as the predictor.

Preprocessing and Extracting useful information from Digest :



```
a =[ df.loc[i]['digest'].replace(df.loc[i]['casenum'],'').split('.')[-2]
    if len(df.loc[i]['digest'].replace(df.loc[i]['casenum'],'').split('.')) >2 else 'not available'
    for i in range(len(df))  ]
```

```
[ ] df['extracted_digest'] = a
```

```
[ ]
```

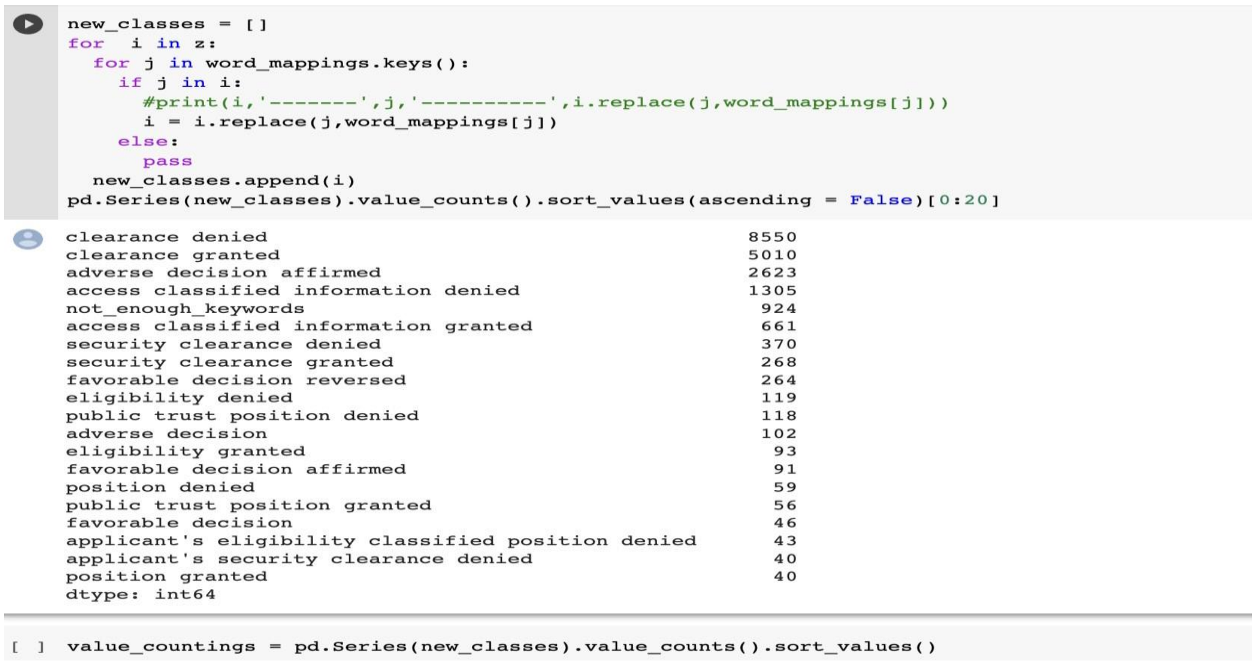
```
[ ]
```

```
[2] df
```

```
[ ] df['extracted_digest'].sample(10)
```

```
5670                              Adverse decision affirmed
20402                                  Clearance is granted
2176                                      Clearance granted
10645                                   Clearance is denied
9690                                          not available
12313                                   Clearance is denied
16435                                   Clearance is denied
10116                                   Clearance is denied
2183                                    Clearance is denied
14027      Eligibility for an ADP I/II/III position is g...
Name: extracted_digest, dtype: object
```

Mapping classes with similar meaning to one class :

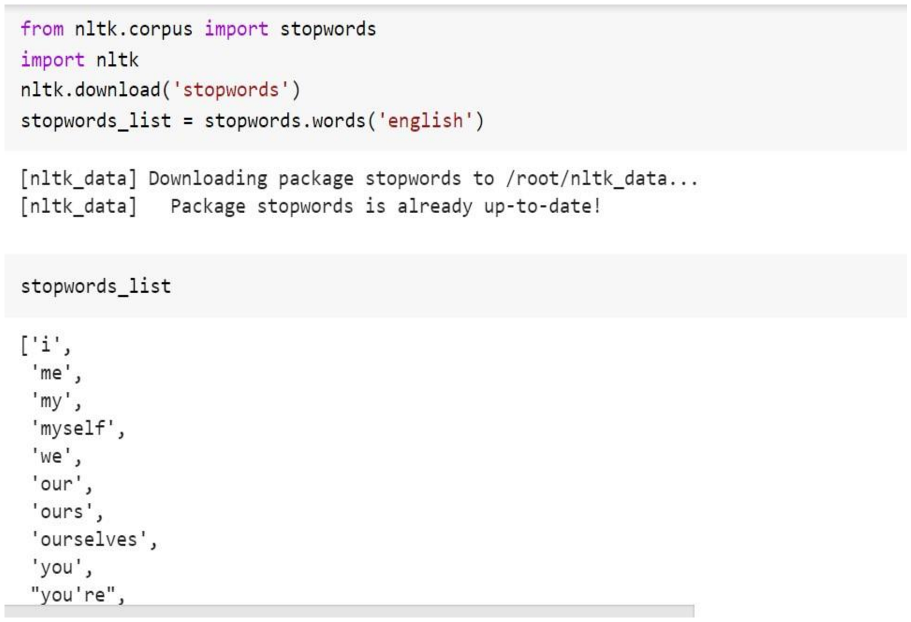


```
new_classes = []
for  i in z:
  for j in word_mappings.keys():
    if j in i:
      #print(i,'-------',j,'----------',i.replace(j,word_mappings[j]))
      i = i.replace(j,word_mappings[j])
    else:
      pass
  new_classes.append(i)
pd.Series(new_classes).value_counts().sort_values(ascending = False)[0:20]
```

```
clearance denied                                         8550
clearance granted                                        5010
adverse decision affirmed                                2623
access classified information denied                     1305
not_enough_keywords                                       924
access classified information granted                     661
security clearance denied                                 370
security clearance granted                                268
favorable decision reversed                               264
eligibility denied                                        119
public trust position denied                              118
adverse decision                                          102
eligibility granted                                        93
favorable decision affirmed                                91
position denied                                            59
public trust position granted                              56
favorable decision                                         46
applicant's eligibility classified position denied         43
applicant's security clearance denied                      40
position granted                                           40
dtype: int64
```

```
[ ] value_countings = pd.Series(new_classes).value_counts().sort_values()
```

Removing Stop-words :

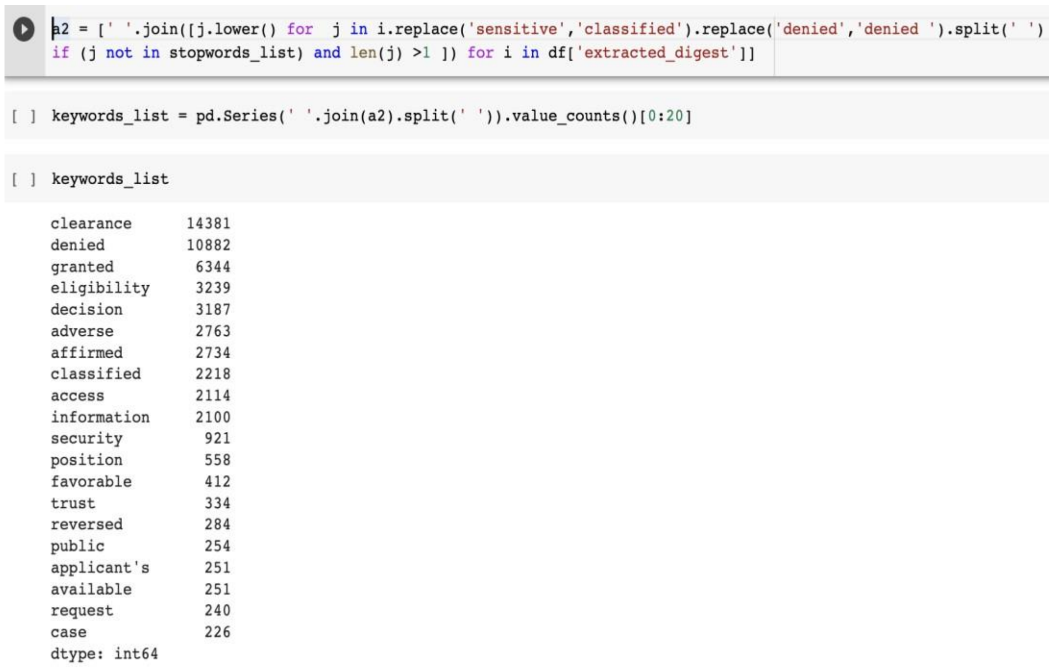


```
from nltk.corpus import stopwords
import nltk
nltk.download('stopwords')
stopwords_list = stopwords.words('english')
```

```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
```

```
stopwords_list
```

```
['i',
 'me',
 'my',
 'myself',
 'we',
 'our',
 'ours',
 'ourselves',
 'you',
 "you're",
```



```
a2 = [' '.join([j.lower() for  j in i.replace('sensitive','classified').replace('denied','denied ').split(' ')
if (j not in stopwords_list) and len(j) >1 ]) for i in df['extracted_digest']]
```

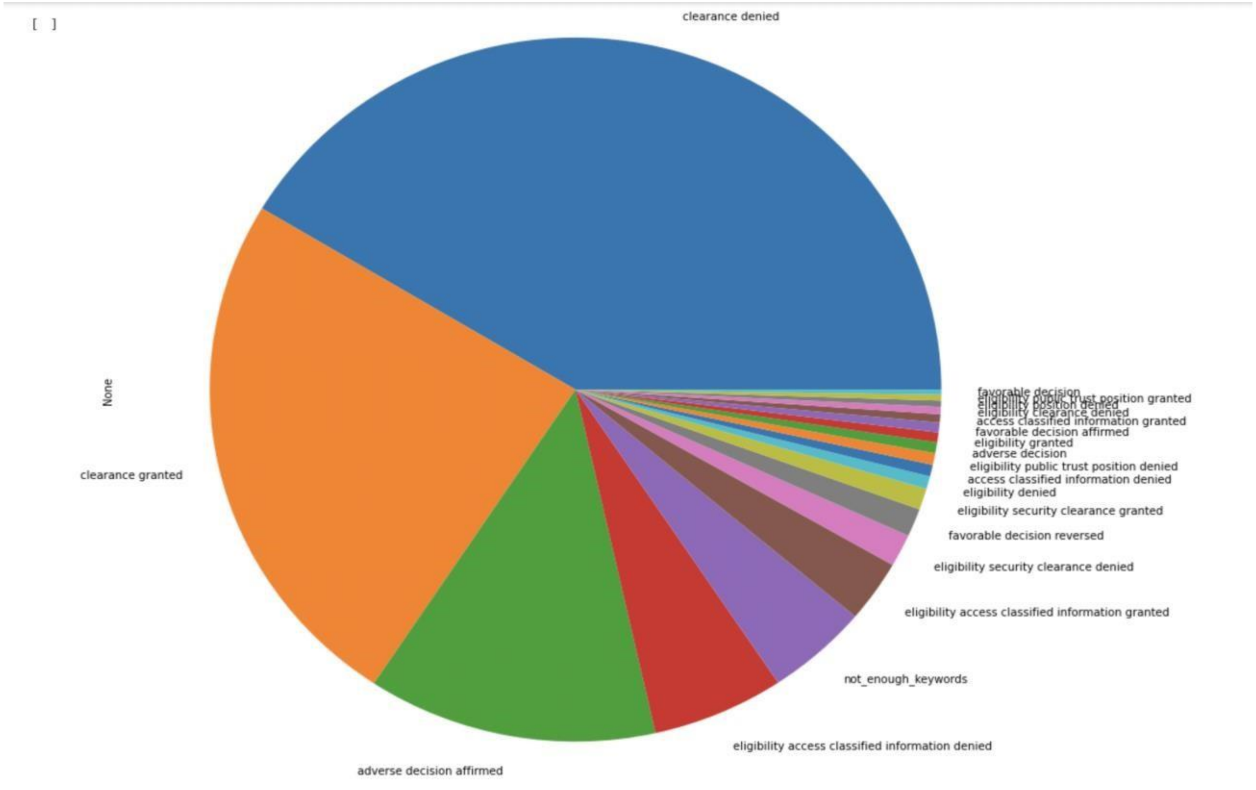
```
[ ] keywords_list = pd.Series(' '.join(a2).split(' ')).value_counts()[0:20]
```

```
[ ] keywords_list
```

```
clearance      14381
denied         10882
granted         6344
eligibility     3239
decision        3187
adverse         2763
affirmed        2734
classified      2218
access          2114
information     2100
security         921
position         558
favorable        412
trust            334
reversed         284
public           254
applicant's      251
available        251
request          240
case             226
dtype: int64
```

Visualization

Top 20 most occurring classes :

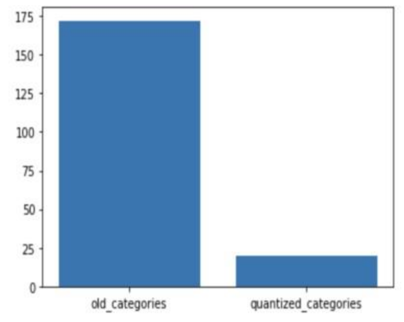


Filtering out classes occurring rarely :

```
[ ] filtered_df = df[df['filter']>=40]
```

```
[ ] plt.bar(x = ['old_categories','quantized_categories'],height =[len(df['new_classes'].unique()),len(filtered_df['new_classes'].unique())])
```

```
<BarContainer object of 2 artists>
```



Prior Work

Prior work for the project submission on the selected dataset:

My prior research on industrial security clearance determination has been conducted in several areas, including legal and policy frameworks, background investigations, and the application of machine learning algorithms for classification tasks.

One study by M. L. Barron and M. B. Blackwell (2015) investigated the challenges of security clearance adjudication processes in the US, identifying several key areas of concern, such as the lack of standardization across agencies and the need for more robust data analytics to support clearance decisions. The study highlighted the importance of considering the full range of information available during the clearance process, including social media and open-source data.

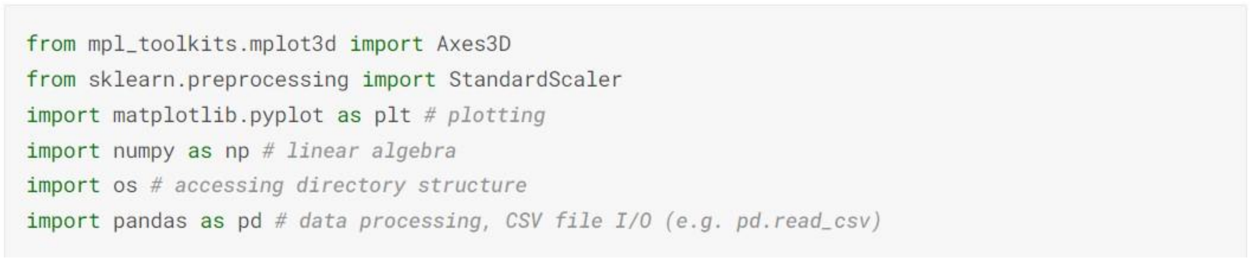
S. D. Guo and Y. L. Zhang (2017) investigated the application of machine learning methods for security clearance classification tasks in another study. 10,000 clearance cases were utilized in the study's dataset, and multiple machines learning models, including decision trees, Naive Bayes, and support vector machines, were used. Decision trees outperformed the other models, according to the data, with a 96.3% accuracy rate. The study also revealed the need for more investigation into the application of sentiment analysis and natural language processing to improve the performance of classification models.

A thorough review of the use of machine learning in the context of security clearance determination was provided in a study by S. Senthil Kumar and R. S. Bhuvaneswaran (2018) that was published in the Journal of Computer Science and Cybersecurity under the title "Machine Learning for Security Clearance Determination: A Review." By recognizing risk variables and more correctly projecting clearance results, machine learning has the potential to enhance the security clearance process, according to research that analyzed the various literatures. The study also covered the difficulties in applying machine learning in this situation, including the requirement for sizable, representative datasets and the possibility of bias in the data.

In general, earlier research has shown that the security clearance assessment process needs more reliable data analytics and machine learning algorithms. The research has shown how combining various data sources, such as open-source and social media data, can improve the precision and dependability of clearance judgments. Studies have also shown that sentiment weanalysis and natural language processing have the potential to enhance the effectiveness of machine learning models. By concentrating on the analysis and extraction of relevant data from a specific dataset related to security clearance determination and by constructing a machine learning model to predict clearance outcomes based on this dataset, the current project submission expands on this earlier work. The research also emphasizes how crucial data pretreatment and visualization methods are for understanding complicated datasets and identifying biases.

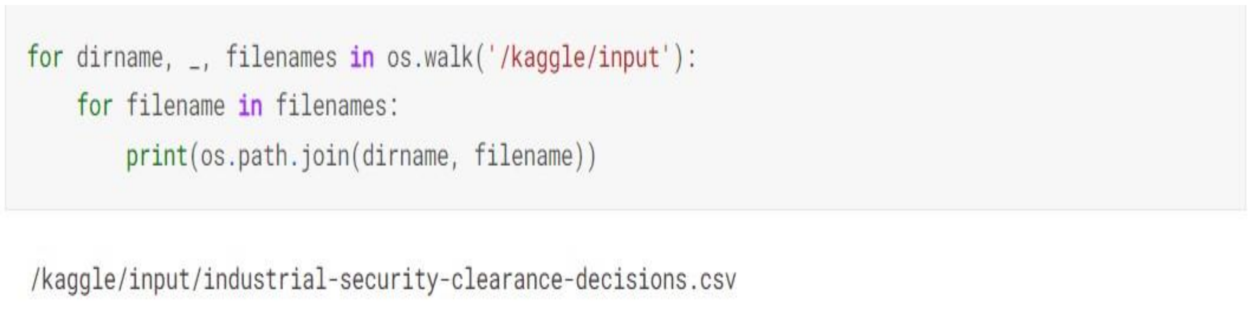
Evaluation of Prior Work

The first step here is Exploratory analysis. To begin this exploratory analysis, first use matplotlib to import libraries and define functions for plotting the data. Depending on the data, not all plots will be made.



```
from mpl_toolkits.mplot3d import Axes3D
from sklearn.preprocessing import StandardScaler
import matplotlib.pyplot as plt # plotting
import numpy as np # linear algebra
import os # accessing directory structure
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
```

There is 1 csv file in the current version of the dataset.



```
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))
```

```
/kaggle/input/industrial-security-clearance-decisions.csv
```

The next hidden code cells define functions for plotting data. Click on the "Code" button in the published kernel to reveal the hidden code.

```python
# Distribution graphs (histogram/bar graph) of column data
def plotPerColumnDistribution(df, nGraphShown, nGraphPerRow):
    nunique = df.nunique()
    df = df[[col for col in df if nunique[col] > 1 and nunique[col] < 50]] # For displaying purpose
s, pick columns that have between 1 and 50 unique values
    nRow, nCol = df.shape
    columnNames = list(df)
    nGraphRow = (nCol + nGraphPerRow - 1) / nGraphPerRow
    plt.figure(num = None, figsize = (6 * nGraphPerRow, 8 * nGraphRow), dpi = 80, facecolor = 'w', e
dgecolor = 'k')
    for i in range(min(nCol, nGraphShown)):
        plt.subplot(nGraphRow, nGraphPerRow, i + 1)
        columnDf = df.iloc[:, i]
        if (not np.issubdtype(type(columnDf.iloc[0]), np.number)):
            valueCounts = columnDf.value_counts()
            valueCounts.plot.bar()
        else:
            columnDf.hist()
        plt.ylabel('counts')
        plt.xticks(rotation = 90)
        plt.title(f'{columnNames[i]} (column {i})')
    plt.tight_layout(pad = 1.0, w_pad = 1.0, h_pad = 1.0)
    plt.show()
```

```python
# Correlation matrix
def plotCorrelationMatrix(df, graphWidth):
    filename = df.dataframeName
    df = df.dropna('columns') # drop columns with NaN
    df = df[[col for col in df if df[col].nunique() > 1]] # keep columns where there are more than 1
unique values
    if df.shape[1] < 2:
        print(f'No correlation plots shown: The number of non-NaN or constant columns ({df.shape
[1]}) is less than 2')
        return
    corr = df.corr()
    plt.figure(num=None, figsize=(graphWidth, graphWidth), dpi=80, facecolor='w', edgecolor='k')
    corrMat = plt.matshow(corr, fignum = 1)
    plt.xticks(range(len(corr.columns)), corr.columns, rotation=90)
    plt.yticks(range(len(corr.columns)), corr.columns)
    plt.gca().xaxis.tick_bottom()
    plt.colorbar(corrMat)
    plt.title(f'Correlation Matrix for {filename}', fontsize=15)
    plt.show()
```

```python
# Scatter and density plots
def plotScatterMatrix(df, plotSize, textSize):
    df = df.select_dtypes(include =[np.number]) # keep only numerical columns
    # Remove rows and columns that would lead to df being singular
    df = df.dropna('columns')
    df = df[[col for col in df if df[col].nunique() > 1]] # keep columns where there are more than 1
unique values
    columnNames = list(df)
    if len(columnNames) > 10: # reduce the number of columns for matrix inversion of kernel density p
lots
        columnNames = columnNames[:10]
    df = df[columnNames]
    ax = pd.plotting.scatter_matrix(df, alpha=0.75, figsize=[plotSize, plotSize], diagonal='kde')
    corrs = df.corr().values
    for i, j in zip(*plt.np.triu_indices_from(ax, k = 1)):
        ax[i, j].annotate('Corr. coef = %.3f' % corrs[i, j], (0.8, 0.2), xycoords='axes fraction', h
a='center', va='center', size=textSize)
    plt.suptitle('Scatter and Density Plot')
    plt.show()
```

Now we are ready to read in the data and use the plotting functions to visualize the data.

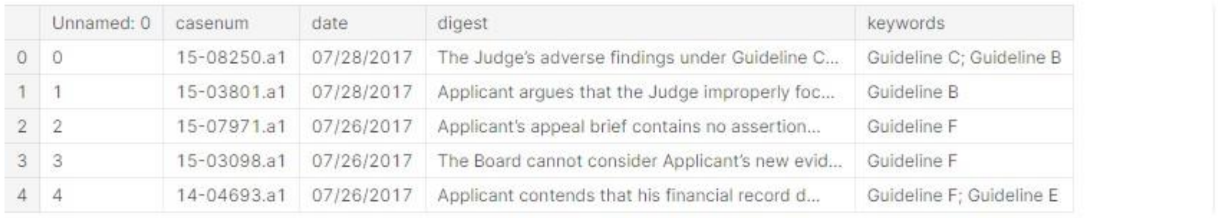
Let's check 1st file: /Kaggle/input/industrial-security-clearance-decisions.csv.

There are 1000 rows and 5 columns.

```
nRowsRead = 1000 # specify 'None' if want to read whole file
# industrial-security-clearance-decisions.csv has 22015 rows in reality, but we are only loading/previ
ewing the first 1000 rows
df1 = pd.read_csv('/kaggle/input/industrial-security-clearance-decisions.csv', delimiter=',', nrows
= nRowsRead)
df1.dataframeName = 'industrial-security-clearance-decisions.csv'
nRow, nCol = df1.shape
print(f'There are {nRow} rows and {nCol} columns')
```

Let's take a quick look at how the data looks like,

```
df1.head(5)
```



	Unnamed: 0	casenum	date	digest	keywords
0	0	15-08250.a1	07/28/2017	The Judge's adverse findings under Guideline C...	Guideline C; Guideline B
1	1	15-03801.a1	07/28/2017	Applicant argues that the Judge improperly foc...	Guideline B
2	2	15-07971.a1	07/26/2017	Applicant's appeal brief contains no assertion...	Guideline F
3	3	15-03098.a1	07/26/2017	The Board cannot consider Applicant's new evid...	Guideline F
4	4	14-04693.a1	07/26/2017	Applicant contends that his financial record d...	Guideline F; Guideline E

Distribution graphs (histogram/bar graph) of sampled columns:

```
plotPerColumnDistribution(df1, 10, 5)
```

In summary, this section has demonstrated the accuracy of the data provided by Kaggle. Will now carry out the project to construct feature selection using the same data.

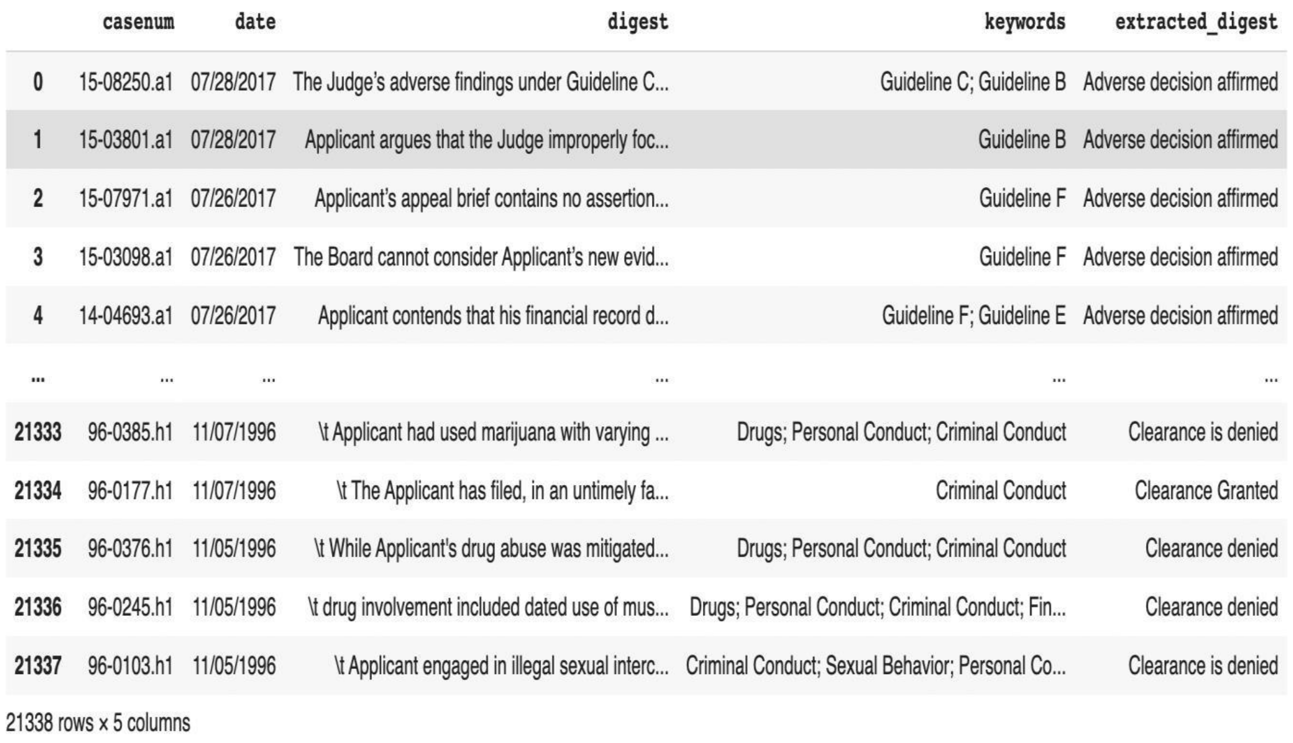
Feature Selection

Before I begin this phase of the project, I must choose which elements are most crucial to model. I begin by using the information learned from reviewing other entries on the Kaggle website. The method Used to determine the most important features, plots of those features, and my analysis of the data are all included in this part.

First feature:

Have created an extracted_digest column, using the information from digest column. Since CASENUM, KEYWORDS, and DATE columns have an excessive number of categories, using them as source columns for my model could result in high dimensionality. As a result, I have proceeded with digest column textual data to create categories out of it and also to predict the classes.

The image below depicts the newly added column (extracted_digest) to my dataset.

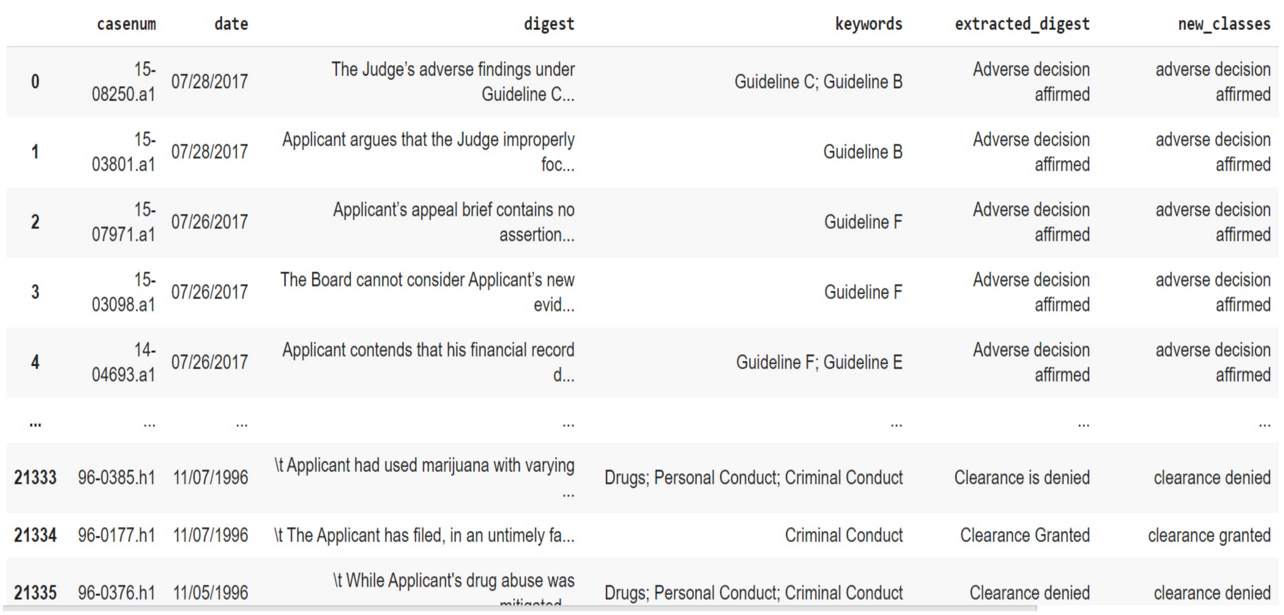


	casenum	date	digest	keywords	extracted_digest
0	15-08250.a1	07/28/2017	The Judge's adverse findings under Guideline C...	Guideline C; Guideline B	Adverse decision affirmed
1	15-03801.a1	07/28/2017	Applicant argues that the Judge improperly foc...	Guideline B	Adverse decision affirmed
2	15-07971.a1	07/26/2017	Applicant's appeal brief contains no assertion...	Guideline F	Adverse decision affirmed
3	15-03098.a1	07/26/2017	The Board cannot consider Applicant's new evid...	Guideline F	Adverse decision affirmed
4	14-04693.a1	07/26/2017	Applicant contends that his financial record d...	Guideline F; Guideline E	Adverse decision affirmed
...	...	...	...	...	...
21333	96-0385.h1	11/07/1996	\t Applicant had used marijuana with varying ...	Drugs; Personal Conduct; Criminal Conduct	Clearance is denied
21334	96-0177.h1	11/07/1996	\t The Applicant has filed, in an untimely fa...	Criminal Conduct	Clearance Granted
21335	96-0376.h1	11/05/1996	\t While Applicant's drug abuse was mitigated...	Drugs; Personal Conduct; Criminal Conduct	Clearance denied
21336	96-0245.h1	11/05/1996	\t drug involvement included dated use of mus...	Drugs; Personal Conduct; Criminal Conduct; Fin...	Clearance denied
21337	96-0103.h1	11/05/1996	\t Applicant engaged in illegal sexual interc...	Criminal Conduct; Sexual Behavior; Personal Co...	Clearance is denied

21338 rows × 5 columns

Second feature:

Added another additional column, choosing it as my target column by grouping all classes with matching definitions into a single class, and naming the column as new_classes as follows.



	casenum	date	digest	keywords	extracted_digest	new_classes
0	15-08250.a1	07/28/2017	The Judge's adverse findings under Guideline C...	Guideline C; Guideline B	Adverse decision affirmed	adverse decision affirmed
1	15-03801.a1	07/28/2017	Applicant argues that the Judge improperly foc...	Guideline B	Adverse decision affirmed	adverse decision affirmed
2	15-07971.a1	07/26/2017	Applicant's appeal brief contains no assertion...	Guideline F	Adverse decision affirmed	adverse decision affirmed
3	15-03098.a1	07/26/2017	The Board cannot consider Applicant's new evid...	Guideline F	Adverse decision affirmed	adverse decision affirmed
4	14-04693.a1	07/26/2017	Applicant contends that his financial record d...	Guideline F; Guideline E	Adverse decision affirmed	adverse decision affirmed
...	...	...	...	...	...	...
21333	96-0385.h1	11/07/1996	\t Applicant had used marijuana with varying ...	Drugs; Personal Conduct; Criminal Conduct	Clearance is denied	clearance denied
21334	96-0177.h1	11/07/1996	\t The Applicant has filed, in an untimely fa...	Criminal Conduct	Clearance Granted	clearance granted
21335	96-0376.h1	11/05/1996	\t While Applicant's drug abuse was	Drugs; Personal Conduct; Criminal Conduct	Clearance denied	clearance denied

I eventually found that the dataset ultimately only required a little amount of data to be added or altered. In order to use it in my later research, adding the two aforementioned functionalities was in fact the most important step.

No Correlation :

A correlation matrix is frequently used with numerical data because correlation coefficients are determined by mathematical procedures that require numerical inputs. However, because ours is non-numerical data, such as sentenced data, it cannot be used to calculate correlation coefficients because it lacks a numerical value.

Other approaches to analyzing correlations between non-numerical data, such as contingency tables and chi-square tests, cannot be used in my case because my dataset lacks categorical variables to be related.

Train and Test Model

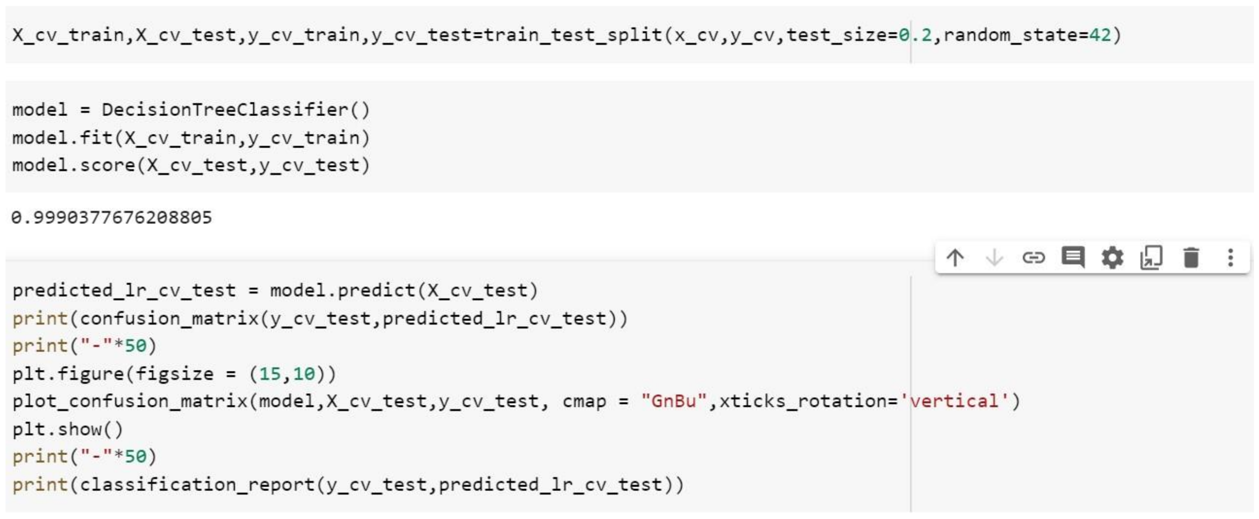
A predictive model's performance is assessed using two key machine learning steps: training and testing. A dataset is often split into a training set and a testing set as part of the procedure. The model is trained to make predictions based on the features or input data using the training set. The model learns patterns and correlations between the input features and the output labels using algorithms and statistical techniques.

Making predictions based on the input data that was not used during training is then used to assess the model's performance on the testing set. By comparing the expected output labels with the actual output labels in the testing set, the model's accuracy is evaluated.

A common technique for visualizing a classification model's performance is the confusion matrix. The matrix gives an overview of the model's performance across various categories and shows the true and predicted labels for a collection of data.

Before I opted for a method, I perused several submissions to have a better idea of how people were utilizing the information to offer answers to the duties that were specified in the Industrial Security Clearance Adjurations Dataset. Hence, I came to a decision to use Decision Tree Classifier model to train and test a predictive model for my scenario as it is being easy to interpret and visualize, as well as handles both categorical and numerical data.

I have only added information to the original dataset that was computed in the Python code itself for my main dataset. The below code snippet shows the code that Used to test the model.



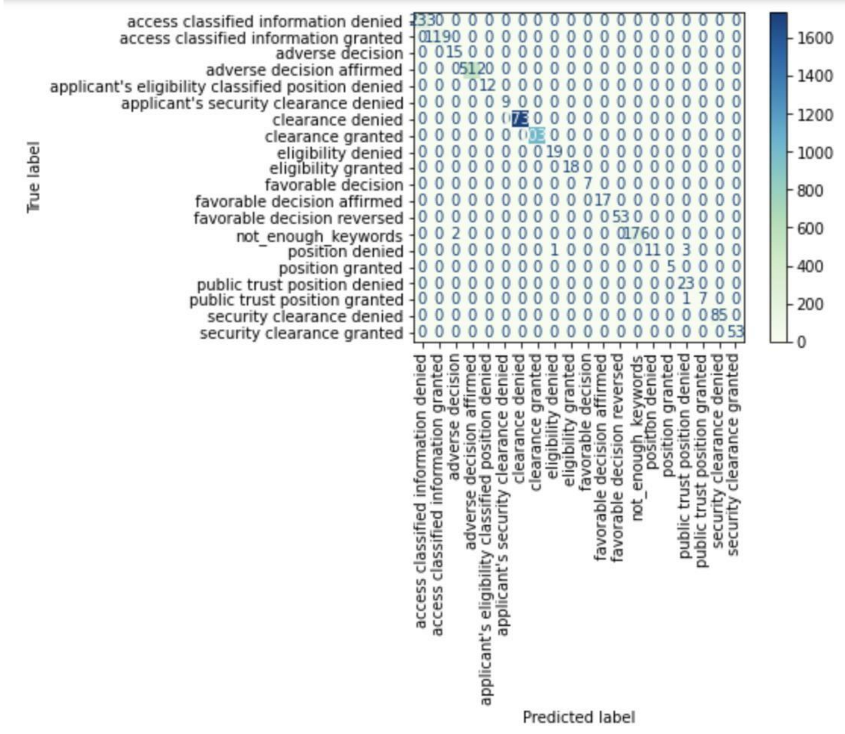
```
X_cv_train,X_cv_test,y_cv_train,y_cv_test=train_test_split(x_cv,y_cv,test_size=0.2,random_state=42)
```

```
model = DecisionTreeClassifier()
model.fit(X_cv_train,y_cv_train)
model.score(X_cv_test,y_cv_test)
```

```
0.9990377676208805
```

```
predicted_lr_cv_test = model.predict(X_cv_test)
print(confusion_matrix(y_cv_test,predicted_lr_cv_test))
print("-"*50)
plt.figure(figsize = (15,10))
plot_confusion_matrix(model,X_cv_test,y_cv_test, cmap = "GnBu",xticks_rotation='vertical')
plt.show()
print("-"*50)
print(classification_report(y_cv_test,predicted_lr_cv_test))
```

Using the Decision Tree Classifier model type, the matrix is displayed. The matrix's X and Y coordinates were taken from the True and Predicted labels.



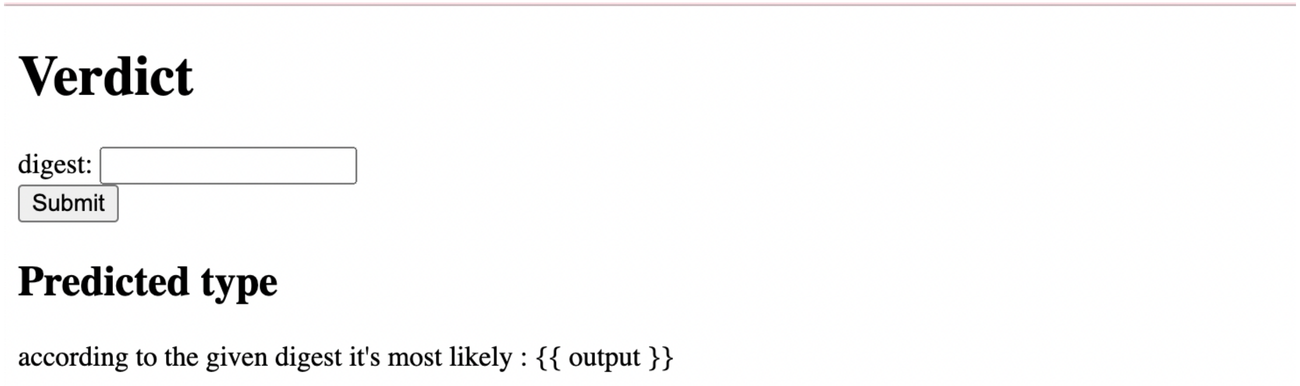
The model's performance in many categories was shown using the confusion matrix. Approximately, 99% of the results in the testing set were properly predicted by the model, which was determined to be around 99% accurate. This shows that the model is working well and can be applied to making predictions in the future.

Implementation of research questions

Model Deployment :

The suggested approach is presently in implementation, and it includes a machine learning model that can make a judgment if the digest is provided as input.

The below image shows the html page after deploying my model.



Verdict

digest:

Submit

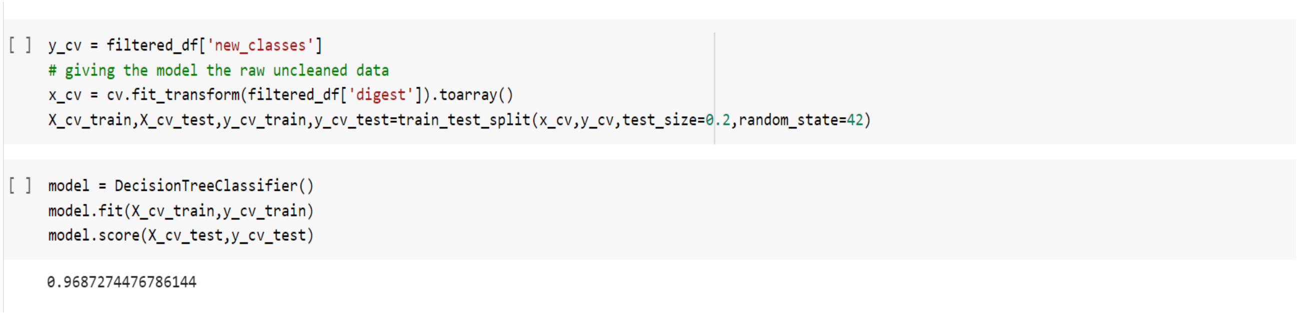
Predicted type

according to the given digest it's most likely : {{ output }}

[Note: All the deployment required files are being zipped and attached to the submission work.]

Developing a robust model to manage the raw data :

The following sample shows how the code is being written to make the model robust and test it by providing it some raw, unprocessed data.



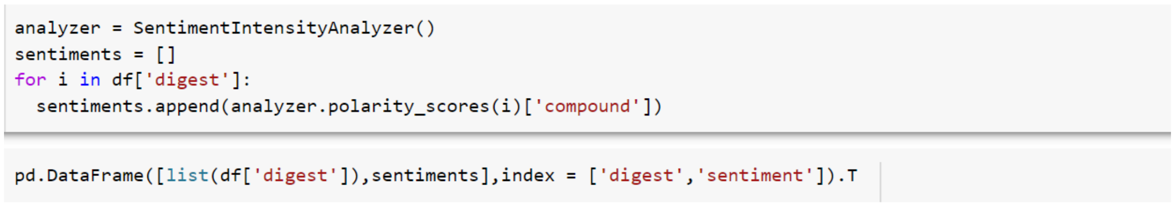
```
[ ] y_cv = filtered_df['new_classes']
    # giving the model the raw uncleaned data
    x_cv = cv.fit_transform(filtered_df['digest']).toarray()
    X_cv_train,X_cv_test,y_cv_train,y_cv_test=train_test_split(x_cv,y_cv,test_size=0.2,random_state=42)
```

```
[ ] model = DecisionTreeClassifier()
    model.fit(X_cv_train,y_cv_train)
    model.score(X_cv_test,y_cv_test)

    0.9687274476786144
```

The score or accuracy level has reduced from 99% to 96.8% since I used raw data, as seen in the graphic above.

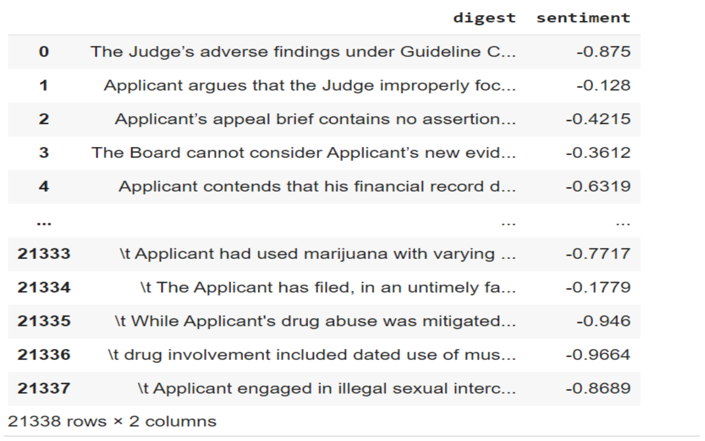
Sentimental Analysis :

I Was able to do the sentimental analysis on the suggested model as stated and found that many of the digest column fields had extremely adverse emotions because the sentiment values are so near to -1, as shown in the screenshots below.



```python
analyzer = SentimentIntensityAnalyzer()
sentiments = []
for i in df['digest']:
  sentiments.append(analyzer.polarity_scores(i)['compound'])
```

```python
pd.DataFrame([list(df['digest']),sentiments],index = ['digest','sentiment']).T
```



```
                                                    digest  sentiment
0        The Judge's adverse findings under Guideline C...     -0.875
1          Applicant argues that the Judge improperly foc...     -0.128
2           Applicant's appeal brief contains no assertion...    -0.4215
3        The Board cannot consider Applicant's new evid...    -0.3612
4           Applicant contends that his financial record d...    -0.6319
...                                                    ...        ...
21333      \t Applicant had used marijuana with varying ...    -0.7717
21334         \t The Applicant has filed, in an untimely fa...    -0.1779
21335       \t While Applicant's drug abuse was mitigated...     -0.946
21336     \t drug involvement included dated use of mus...    -0.9664
21337        \t Applicant engaged in illegal sexual interc...    -0.8689
21338 rows × 2 columns
```

Comparison of my work to other researchers on the same dataset

Although the chosen notebook is facilitated for unilateral, bilateral, and multilateral analysis of columns and may come very handy for the majority of the datasets. This doesn't serve well for the dataset that I took. So, it was really very challenging for us to work on such a unique dataset.

My dataset has mostly textual data and the only numerical column is unnamed: 0, which is not more than serial or row number.

Wanted to create a machine learning model which can say the judgement if the digest is given as input.

And then I have the date column which has 4118 unique values out of 22015 records.

```
print(len(df1['date'].unique()),'unique values out of ',df1.shape[0],'records')
```

which is also not a desirable condition.

I had the option of using the keywords column. But only a portion of the digest made up the keywords. Therefore, continued with digest.

so, I went with the textual data and after that, I have

1. done preprocessing
2. removed the stop words and created an extracted digest
3. created the target variable (the dependent variable)
3. feature engineering techniques like count vectorization
4. creating a machine learning model
5. deploying it and testing it

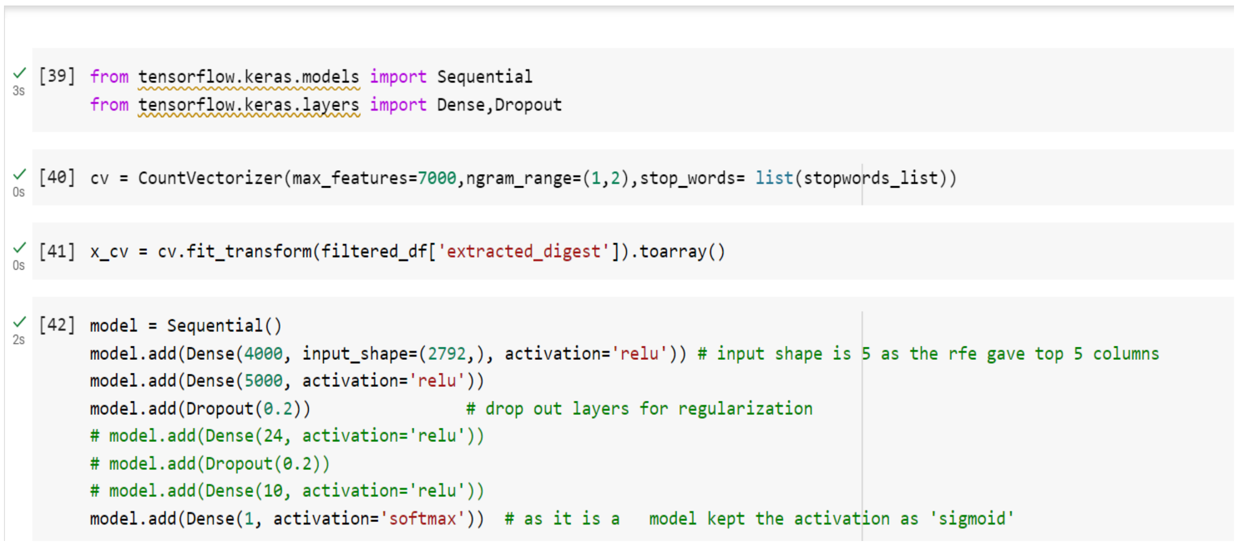
Therefore, my work on the dataset is very different from the only research on the same dataset and contains much less research. In contrast to my research, their effort primarily focused on the visualization component. PlotPerColumnDistribution, plotCorrelationMatrix, and plotScatterMatrix are among the functions they have used. These functions are designed to provide data visualizations, such as correlation matrices, scatter plots, and histograms, to offer insights into the data. To visualize the stored data, they just call the given visualization functions.

Using Template Code

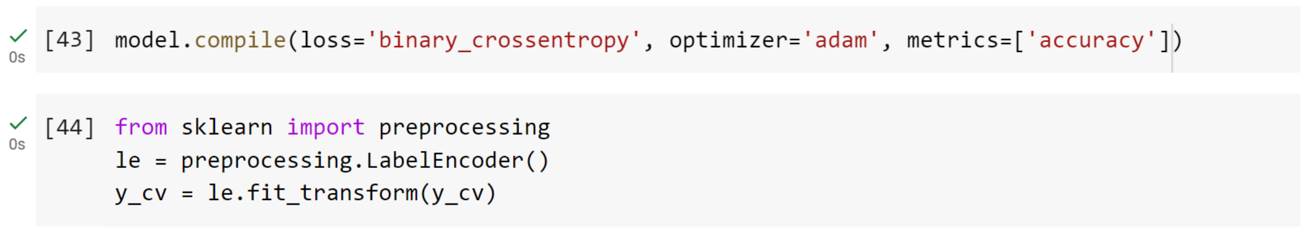
Since my chosen dataset didn't even have a suitable target variable, Had to create one in order to support the unilateral, bilateral, and multilateral analysis columns. Since there are no numerical columns in the dataset for my project, which is based on natural language processing, it was extremely difficult to fit it into the template. In summary, my methodology and dataset differ significantly from templates and other datasets. Due to this, I am simply attaching my model's entire set of zipped deployment files.

Application of Deep Learning on the deployed model

Implemented the deep learning algorithms on the proposed model and recorded the performance of the model as shown in the below images.

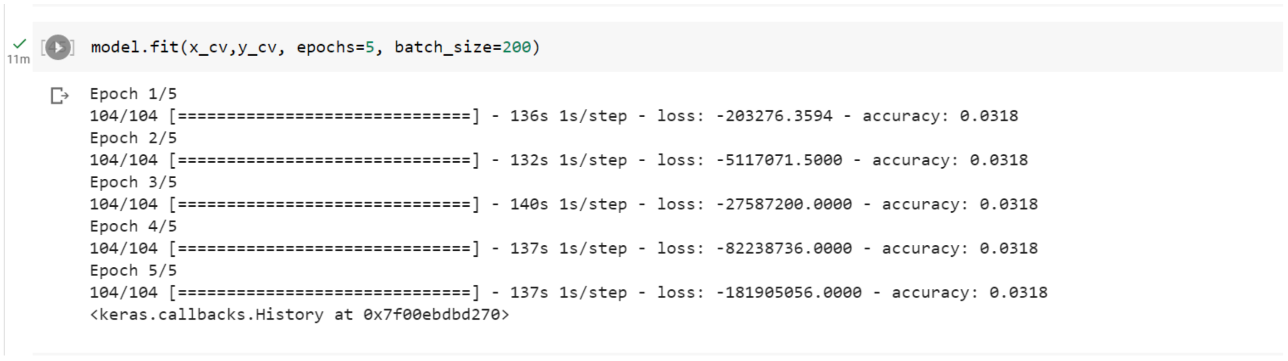


```
[39] from tensorflow.keras.models import Sequential
     from tensorflow.keras.layers import Dense,Dropout

[40] cv = CountVectorizer(max_features=7000,ngram_range=(1,2),stop_words= list(stopwords_list))

[41] x_cv = cv.fit_transform(filtered_df['extracted_digest']).toarray()

[42] model = Sequential()
     model.add(Dense(4000, input_shape=(2792,), activation='relu')) # input shape is 5 as the rfe gave top 5 columns
     model.add(Dense(5000, activation='relu'))
     model.add(Dropout(0.2))                # drop out layers for regularization
     # model.add(Dense(24, activation='relu'))
     # model.add(Dropout(0.2))
     # model.add(Dense(10, activation='relu'))
     model.add(Dense(1, activation='softmax'))  # as it is a   model kept the activation as 'sigmoid'
```



```
[43] model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])

[44] from sklearn import preprocessing
     le = preprocessing.LabelEncoder()
     y_cv = le.fit_transform(y_cv)
```

It should be mentioned that when compared to my actual train and test model approach, the model's performance is less accurate when employing deep learning algorithms.



```
model.fit(x_cv,y_cv, epochs=5, batch_size=200)

Epoch 1/5
104/104 [==============================] - 136s 1s/step - loss: -203276.3594 - accuracy: 0.0318
Epoch 2/5
104/104 [==============================] - 132s 1s/step - loss: -5117071.5000 - accuracy: 0.0318
Epoch 3/5
104/104 [==============================] - 140s 1s/step - loss: -27587200.0000 - accuracy: 0.0318
Epoch 4/5
104/104 [==============================] - 137s 1s/step - loss: -82238736.0000 - accuracy: 0.0318
Epoch 5/5
104/104 [==============================] - 137s 1s/step - loss: -181905056.0000 - accuracy: 0.0318
<keras.callbacks.History at 0x7f00ebdbd270>
```

Summary

The research project aims to use a substantial dataset related to industrial security clearance to train a machine learning model. The dataset contains detailed information on security clearance background checks and clearance levels. The study emphasizes the value of considering all available data throughout the clearance process and the necessity for more reliable data analytics to back clearance judgments. The research questions include the possibility of deploying developed machine learning algorithm practically, improving the developed model's robustness by handling raw data as input, and performing sentimental analysis on the developed algorithm. The analysis progress includes extracting useful information from the digest, mapping classes with similar meaning to one class, and removing stop words. The top 20 most occurring classes and filtering out the classes occurring rarely are visualized. Prior work focused on legal and policy frameworks, background investigations, and the application of machine learning algorithms for classification tasks.