

```
ILOM 2KTEGLU: MONGET PETECCTON TMBOLC CLGTH CESC PATTO
47
        from sklearn.feature extraction.text import CountVectorizer
48
        from sklearn.linear_model import LogisticRegression
49
        from sklearn.metrics import roc_auc_score
50
        from sklearn.metrics import confusion_matrix
51
52
        # %% [markdown]
53
        # *Import the data & Clean ups*
54
        # %% [code] {"execution":{"iopub.status.busy":"2023-10-24T21:20:12.47781Z","iopub.execute_input":"2023-10-24T
55
        #importing data
56
57
        fake_data = pd.read_csv('/kaggle/input/fake-and-real-news-dataset/Fake.csv')
        print("fake_data",fake_data.shape)
58
59
        true_data= pd.read_csv('/kaggle/input/fake-and-real-news-dataset/True.csv')
60
        print("true_data",true_data.shape)
61
62
63
        # %% [code] {"execution":{"iopub.status.busy":"2023-10-24T21:20:14.043858Z","iopub.execute_input":"2023-10-24
64
        fake data.head(5)
65
66
        # %% [code] {"execution":{"iopub.status.busy":"2023-10-24T21:20:14.066765Z","iopub.execute_input":"2023-10-24
        true data.head(5)
67
69
        # %% [code] {"execution":{"iopub.status.busy":"2023-10-24T21:20:14.0866Z","iopub.execute_input":"2023-10-24T2
70
        #adding additional column to seperate betwee true & fake data
        # true =1, fake =0
71
72
        true data['target'] = 1
73
        fake_data['target'] = 0
74
        df = pd.concat([true_data, fake_data]).reset_index(drop = True)
75
        df['original'] = df['title'] + ' ' + df['text']
        df.head()
76
77
78
        # %% [code] {"execution":{"iopub.status.busy":"2023-10-24T21:20:14.737583Z","iopub.execute_input":"2023-10-24
79
        df.isnull().sum()
80
81
        # %% [markdown]
        # *Data Clean up*
82
83
        # - create a function here that will be responsible to remove any unneccesary words (Stopwords) from the data
24
85
        # %% [code] {"execution":{"iopub.status.busy":"2023-10-24T21:20:14.781215Z","iopub.execute input":"2023-10-24
86
        stop_words = stopwords.words('english')
87
        stop_words.extend(['from', 'subject', 're', 'edu', 'use'])
88 ∨ def preprocess(text):
89
            result = []
            for token in gensim.utils.simple_preprocess(text):
90
91
                if token not in gensim.parsing.preprocessing.STOPWORDS and len(token) > 2 and token not in stop_words
92
                    result.append(token)
93
94
            return result
95
96
        # %% [code] {"execution":{"iopub.status.busy":"2023-10-24T21:20:14.79379Z","iopub.execute input":"2023-10-24T
97
        # Transforming the unmatching subjects to the same notation
        df.subject=df.subject.replace({'politics':'PoliticsNews','politicsNews':'PoliticsNews'})
98
99
        # %% [code] {"execution":{"iopub.status.busy":"2023-10-24T21:20:14.819181Z","iopub.execute_input":"2023-10-24
100
        sub_tf_df=df.groupby('target').apply(lambda x:x['title'].count()).reset_index(name='Counts')
101
        sub_tf_df.target.replace({0:'False',1:'True'},inplace=True)
102
103
        fig = px.bar(sub_tf_df, x="target", y="Counts",
                     color='Counts', barmode='group',
104
                     height=350)
105
106
        fig.show()
107
```

```
108
        # %% [markdown]
109
        # - The data looks balanced and no issues on building the model
110
        # %% [code] {"execution":{"iopub.status.busy":"2023-10-24T21:20:14.938929Z","iopub.execute_input":"2023-10-24
111
112
        sub_check=df.groupby('subject').apply(lambda x:x['title'].count()).reset_index(name='Counts')
        fig=px.bar(sub_check,x='subject',y='Counts',color='Counts',title='Count of News Articles by Subject')
113
114
        fig.show()
115
116
        # %% [markdown]
117
118
119
        # %% [code] {"execution":{"iopub.status.busy":"2023-10-24T21:20:15.056981Z","iopub.execute_input":"2023-10-24
120
        df['clean_title'] = df['title'].apply(preprocess)
        df['clean_title'][0]
121
122
123
        # %% [code] {"execution":{"iopub.status.busy":"2023-10-24T21:20:18.393978Z","iopub.execute_input":"2023-10-24
        df['clean_joined_title']=df['clean_title'].apply(lambda x:" ".join(x))
124
125
        # %% [code] {"execution":{"iopub.status.busy":"2023-10-24T21:20:18.444948Z","iopub.execute_input":"2023-10-24
126
        plt.figure(figsize = (20,20))
127
128
        wc = WordCloud(max_words = 2000 , width = 1600 , height = 800 , stopwords = stop_words).generate(" ".join(df[
        plt.imshow(wc, interpolation = 'bilinear')
129
130
        # %% [code] {"execution":{"iopub.status.busy":"2023-10-24T21:20:40.057751Z","iopub.execute_input":"2023-10-24
131
        maxlen = -1
132
        for doc in df.clean_joined_title:
133
            tokens = nltk.word_tokenize(doc)
135
            if(maxlen<len(tokens)):</pre>
136
                maxlen = len(tokens)
        print("The maximum number of words in a title is =", maxlen)
137
138
        fig = px.histogram(x = [len(nltk.word_tokenize(x)) for x in df.clean_joined_title], nbins = 50)
139
        fig.show()
140
141
        # %% [markdown]
        # *Creating Prediction Model*
142
143
        # %% [code] {"execution":{"iopub.status.busy":"2023-10-24T21:20:57.606819Z","iopub.execute_input":"2023-10-24
144
        X_train, X_test, y_train, y_test = train_test_split(df.clean_joined_title, df.target, test_size = 0.2,random_
146
        vec_train = CountVectorizer().fit(X_train)
147
        X_vec_train = vec_train.transform(X_train)
148
        X_vec_test = vec_train.transform(X_test)
149
150
        # %% [code] {"execution":{"iopub.status.busy":"2023-10-24T21:20:59.134487Z","iopub.execute_input":"2023-10-24
151
        #model
152
        model = LogisticRegression(C=2)
153
154
        #fit the model
155
        model.fit(X_vec_train, y_train)
156
        predicted_value = model.predict(X_vec_test)
157
158
        #accuracy & predicted value
159
        accuracy_value = roc_auc_score(y_test, predicted_value)
160
        print(accuracy value)
161
162
        # %% [markdown]
163
        # *Create the confusion matrix*
164
165
        # %% [code] {"execution":{"iopub.status.busy":"2023-10-24T21:21:01.621433Z","iopub.execute_input":"2023-10-24
166
        cm = confusion_matrix(list(y_test), predicted_value)
        plt.figure(figsize = (7, 7))
167
168
        sns.heatmap(cm, annot = True,fmt='g',cmap='viridis')
```

```
170
        # %% [markdown]
171
        # - 4465 Fake News have been Classified as Fake
        # - 4045 Real News have been classified as Real
172
173
174
        # %% [markdown]
175
        # *Checking the content of news*
176
177
        # %% [code] {"execution":{"iopub.status.busy":"2023-10-24T21:21:02.055182Z","iopub.execute_input":"2023-10-24
178
        df['clean_text'] = df['text'].apply(preprocess)
179
        df['clean_joined_text']=df['clean_text'].apply(lambda x:" ".join(x))
180
        # %% [code] {"execution":{"iopub.status.busy":"2023-10-24T21:22:28.723648Z","iopub.execute_input":"2023-10-24
181
182
        plt.figure(figsize = (20,20))
183
        wc = WordCloud(max_words = 2000 , width = 1600 , height = 800 , stopwords = stop_words).generate(" ".join(df[
184
        plt.imshow(wc, interpolation = 'bilinear')
185
        # %% [code] {"execution":{"iopub.status.busy":"2023-10-24T21:23:30.622781Z","iopub.execute_input":"2023-10-24
186
187
        maxlen = -1
        for doc in df.clean_joined_text:
188
189
            tokens = nltk.word_tokenize(doc)
            if(maxlen<len(tokens)):</pre>
190
191
                maxlen = len(tokens)
        print("The maximum number of words in a News Content is =", maxlen)
192
193
        fig = px.histogram(x = [len(nltk.word_tokenize(x)) for x in df.clean_joined_text], nbins = 50)
194
        fig.show()
195
196
        # %% [markdown]
197
        # *Predicting the Model*
198
199
        # %% [code] {"execution":{"iopub.status.busy":"2023-10-24T21:27:23.232691Z","iopub.execute_input":"2023-10-24
200
        X_train, X_test, y_train, y_test = train_test_split(df.clean_joined_text, df.target, test_size = 0.2, random_s
201
        vec_train = CountVectorizer().fit(X_train)
        X_vec_train = vec_train.transform(X_train)
203
204
        X_vec_test = vec_train.transform(X_test)
205
        model = LogisticRegression(C=2.5)
206
        model.fit(X_vec_train, y_train)
207
        predicted_value = model.predict(X_vec_test)
        accuracy_value = roc_auc_score(y_test, predicted_value)
208
209
        print(accuracy_value)
210
211
        # %% [code] {"execution":{"iopub.status.busy":"2023-10-24T21:30:15.444444Z","iopub.execute_input":"2023-10-24
212
        prediction = []
        for i in range(len(predicted_value)):
213
214
            if predicted_value[i].item() > 0.5:
215
                prediction.append(1)
216
            else:
217
                prediction.append(0)
218
        cm = confusion_matrix(list(y_test), prediction)
219
        plt.figure(figsize = (6, 6))
220
        sns.heatmap(cm, annot = True,fmt='g')
221
222
        # %% [code]
```