US Election

September 10, 2020

[1]:
```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

def read_data(csv_file):
    try:
        return pd.read_csv(csv_file)
    except:
        print("The file is not found")
        return None


US_election_data_set = read_data("C:/Users/omri1/PycharmProjects/untitled2/
 ↪primary_results.csv")
```

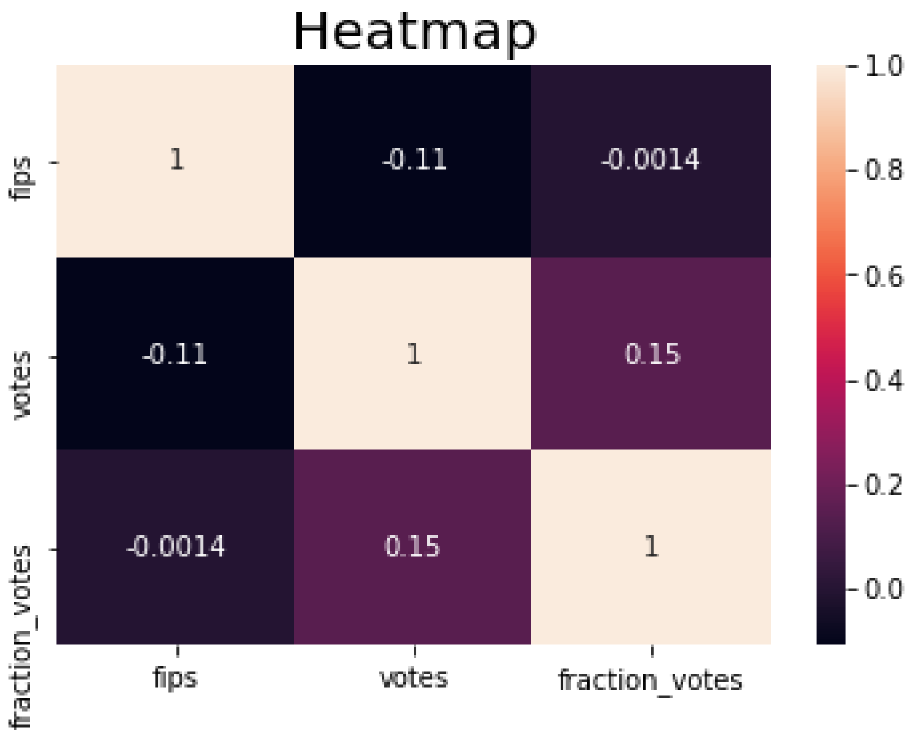
[5]:
```python
US_election_data_set
```

[5]:
```
         state state_abbreviation          county         fips       party  \
0      Alabama                 AL         Autauga       1001.0    Democrat
1      Alabama                 AL         Autauga       1001.0    Democrat
2      Alabama                 AL         Baldwin       1003.0    Democrat
3      Alabama                 AL         Baldwin       1003.0    Democrat
4      Alabama                 AL         Barbour       1005.0    Democrat
…        …                  …             …           …          …
24606  Wyoming                 WY  Teton-Sublette   95600028.0  Republican
24607  Wyoming                 WY   Uinta-Lincoln   95600027.0  Republican
24608  Wyoming                 WY   Uinta-Lincoln   95600027.0  Republican
24609  Wyoming                 WY   Uinta-Lincoln   95600027.0  Republican
24610  Wyoming                 WY   Uinta-Lincoln   95600027.0  Republican

             candidate  votes  fraction_votes
0       Bernie Sanders    544           0.182
1      Hillary Clinton   2387           0.800
2       Bernie Sanders   2694           0.329
3      Hillary Clinton   5290           0.647
4       Bernie Sanders    222           0.078
```

```
…                  …     …             …
24606         Ted Cruz      0           0.000
24607     Donald Trump      0           0.000
24608      John Kasich      0           0.000
24609      Marco Rubio      0           0.000
24610         Ted Cruz     53           1.000

[24611 rows x 8 columns]
```

[2]:
```python
def data_shape(data, label):
    print('Rows number of ' + label + " is: ", data.shape[0])
    print('Columns number of ' + label + ' is: ', data.shape[1])

def data_columns(data):
    return list(data.columns)

def describe_data(data):
    return data.describe()

data_shape(US_election_data_set, 'US election data set')
data_columns(US_election_data_set)
describe_data(US_election_data_set)

sns.heatmap(US_election_data_set.corr(), annot = True)
plt.title('Heatmap', fontsize = 20)
plt.show()
```

```
Rows number of US election data set is:  24611
Columns number of US election data set is:  8
```



[ ]:
```
# Don't have a serious relationship between the numerical columns. It can be␣
↪seen that the votes column affects the fraction vote in a positive way.
```

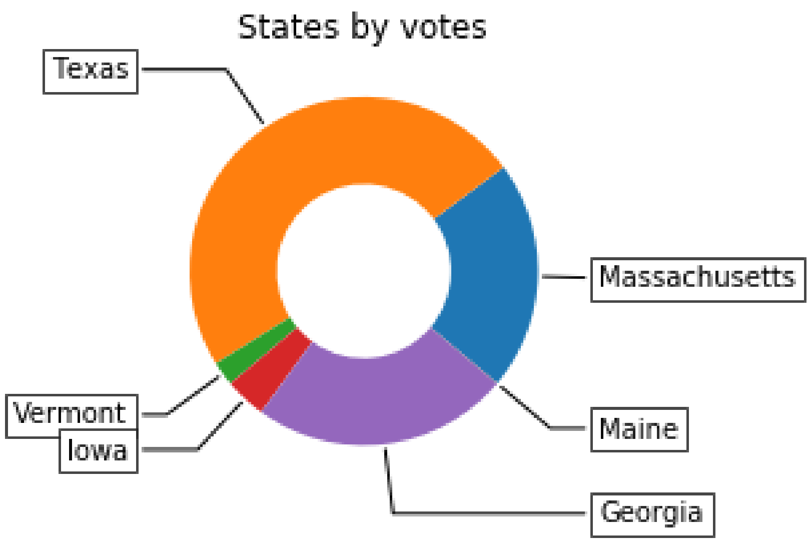
[7]:
```
US_election_data_set['state'].value_counts().head(6)
```

[7]:
```
Massachusetts    2808
Texas            1778
Vermont          1722
Iowa             1485
Georgia          1113
Maine             994
Name: state, dtype: int64
```

[ ]:
```
# Most of the counties came from Massachusetts.
```

[8]:
```
def visualize_data_by_votes(data_frame, label):
    Massachusetts = data_frame[data_frame['state'] == 'Massachusetts']['votes'].
↪sum()
    Texas = data_frame[data_frame['state'] == 'Texas']['votes'].sum()
    Vermont = data_frame[data_frame['state'] == 'Vermont']['votes'].sum()
    Iowa = data_frame[data_frame['state'] == 'Iowa']['votes'].sum()
    Georgia = data_frame[data_frame['state'] == 'Georgia']['votes'].sum()
    Maine = data_frame[data_frame['state'] == 'Maine']['votes'].sum()
```

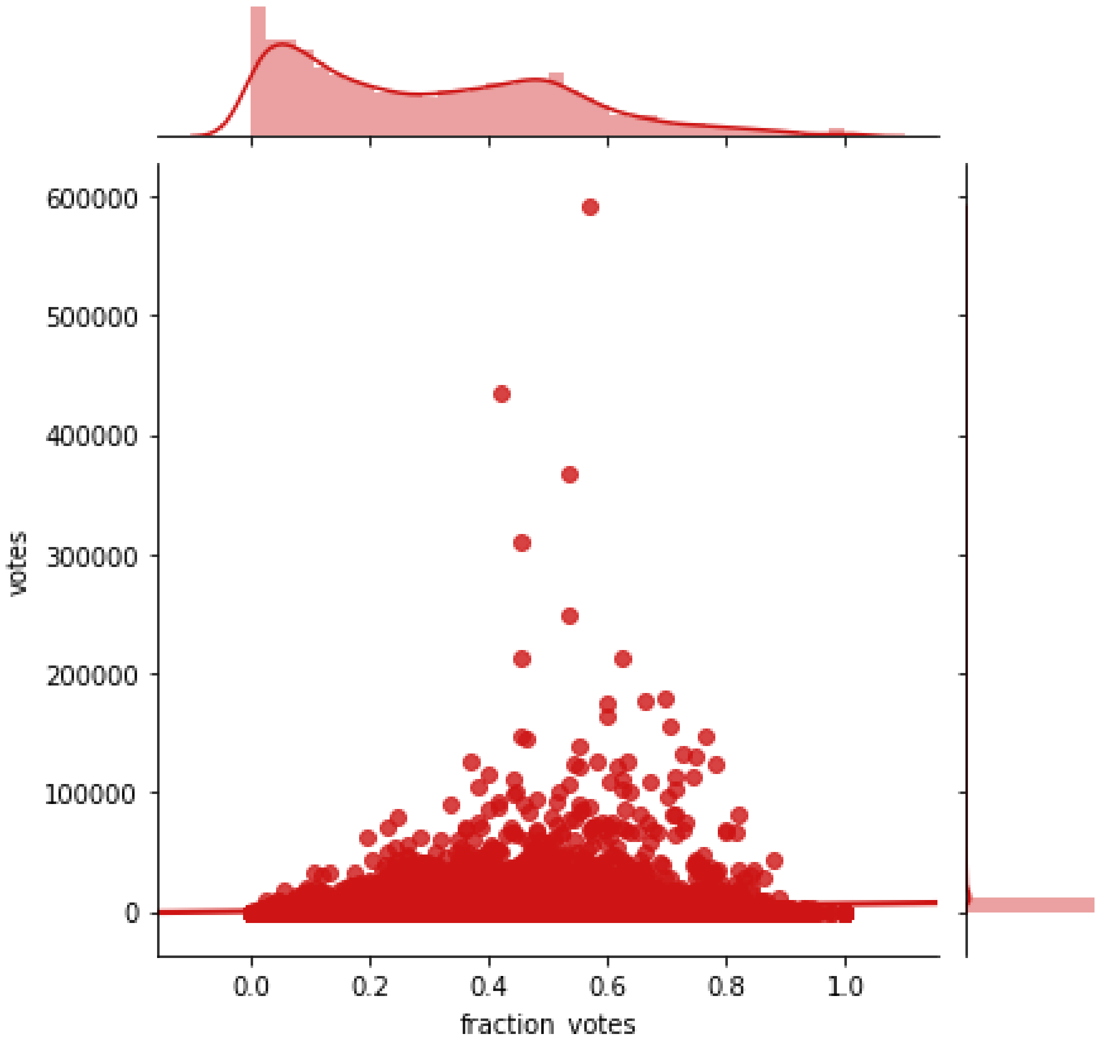
```
    fig, ax = plt.subplots(figsize=(6, 3), subplot_kw=dict(aspect="equal"))

    months = ['Massachusetts',
              'Texas',
              'Vermont',
              'Iowa',
              'Georgia',
              'Maine']

    data = [Massachusetts, Texas, Vermont, Iowa, Georgia, Maine]

    wedges, texts = ax.pie(data, wedgeprops=dict(width=0.5), startangle=-40)

    bbox_props = dict(boxstyle="square,pad=0.3", fc="w", ec="k", lw=0.72)
    kw = dict(arrowprops=dict(arrowstyle="-"),
              bbox=bbox_props, zorder=0, va="center")

    for i, p in enumerate(wedges):
        ang = (p.theta2 - p.theta1) / 2. + p.theta1
        y = np.sin(np.deg2rad(ang))
        x = np.cos(np.deg2rad(ang))
        horizontalalignment = {-1: "right", 1: "left"}[int(np.sign(x))]
        connectionstyle = "angle,angleA=0,angleB={}".format(ang)
        kw["arrowprops"].update({"connectionstyle": connectionstyle})
        ax.annotate(months[i], xy=(x, y), xytext=(1.35 * np.sign(x), 1.4 * y),
                    horizontalalignment=horizontalalignment, **kw)

    ax.set_title(label)

    plt.show()

visualize_data_by_votes(US_election_data_set, "States by votes")
```




```
[ ]: # although most of the counties are from Massachusetts, most of the votes are␣
     ↪from Texas. It's mean that in the population of Texas the percentage vote is␣
     ↪pretty high.
```

```
[9]: import seaborn as sns
     import matplotlib.pyplot as plt
     import time
     from sklearn.model_selection import train_test_split
     from sklearn.ensemble import RandomForestClassifier
     from sklearn.metrics import f1_score,confusion_matrix
     from sklearn.metrics import accuracy_score
     from sklearn.feature_selection import SelectKBest
     from sklearn.feature_selection import chi2
     from sklearn.feature_selection import RFE

     sns.jointplot(US_election_data_set['fraction_votes'],␣
     ↪US_election_data_set['votes'], kind="regg", color="#ce1414")
```

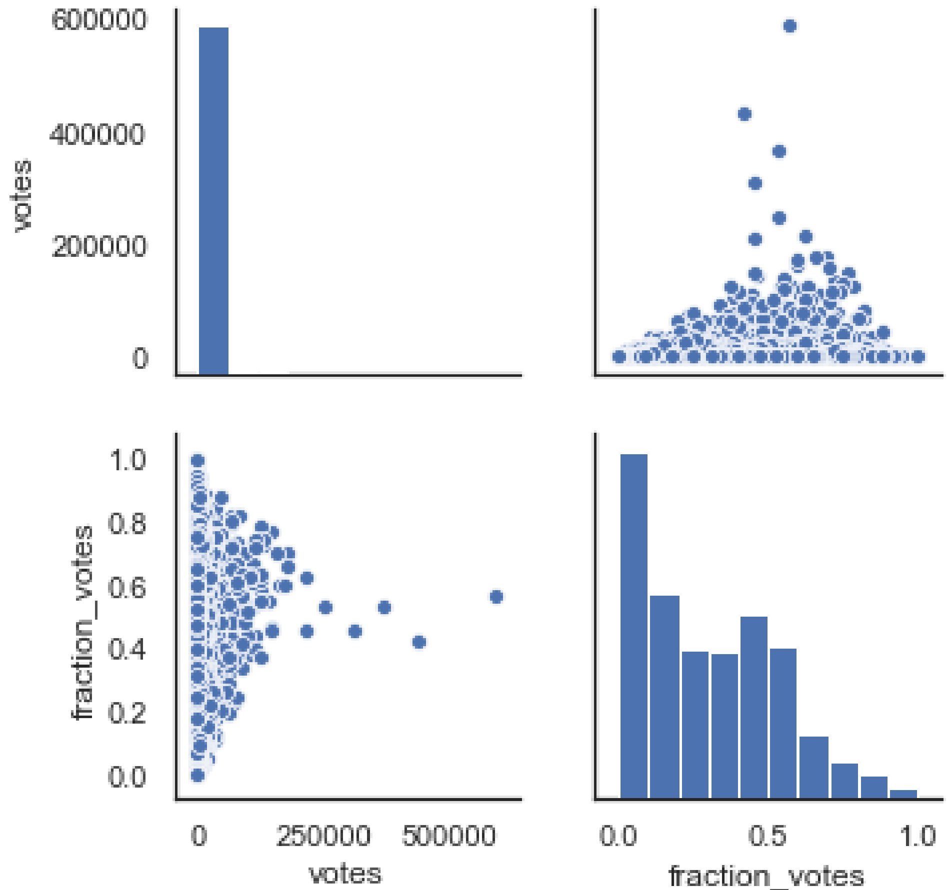
```
[9]: <seaborn.axisgrid.JointGrid at 0x2d4fb147198>
```



[ ]:
```
# According to the graph, it can be seen the number of votes has a positive␣
↪effection on the 'fraction_votes' until some point from there it's going␣
↪down. It can be concluded, in big counties, the percentage vote is decreased.
```

[10]:
```
sns.set(style="white")
sns.pairplot(US_election_data_set[['votes','fraction_votes']])
```

[10]: <seaborn.axisgrid.PairGrid at 0x2d4fc59f978>



[ ]:
```
# These plots confirm the conclusion from the previous graph.
```

[20]:
```
from sklearn import preprocessing
import pandas as pd
import numpy as np

def pre_processing(original_data, supervised_or_unsupervised):
    def remove_columns(data_frame, col_name):
        try:
            columns = list(data_frame.columns)
            include_columns = [x for x in columns if x not in col_name]
            new_data_frame = data_frame[include_columns]
            return new_data_frame
        except:
            print('Something got wrong - remove_columns')

    name = original_data.isnull().sum().where(lambda x: x > 2500).dropna().
 ↪keys().to_list()
```

```
    original_data = remove_columns(original_data, name)

    original_data = remove_columns(original_data, 'state')
    original_data = remove_columns(original_data, 'fips')

    if supervised_or_unsupervised == 'supervised':
        data_frame_for_supervised = original_data

        def encoders(data_frame, label):
            try:
                encoders = {
                    label: preprocessing.LabelEncoder()
                }
                data_frame[label] = encoders[label].
↪fit_transform(data_frame[label].astype(str))
            except:
                print('Something got wrong - encoders')

        encoders(data_frame_for_supervised, 'party')
        encoders(data_frame_for_supervised, 'county')
        encoders(data_frame_for_supervised, 'state_abbreviation')
        encoders(data_frame_for_supervised, 'candidate')

        return data_frame_for_supervised

    if supervised_or_unsupervised == 'unsupervised':
        data_frame_for_unsupervised = original_data

        def get_dummies(data_frame):
            try:
                data_frame = pd.get_dummies(data_frame)
                return data_frame
            except:
                print('Something got wrong - get_dummies')

        data_frame_for_unsupervised = get_dummies(data_frame_for_unsupervised)

        return data_frame_for_unsupervised

    return original_data


US_election_data_set_for_supervised = pre_processing(US_election_data_set,␣
↪'supervised')

US_election_data_set_for_unsupervised = pre_processing(US_election_data_set,␣
↪'unsupervised')
```

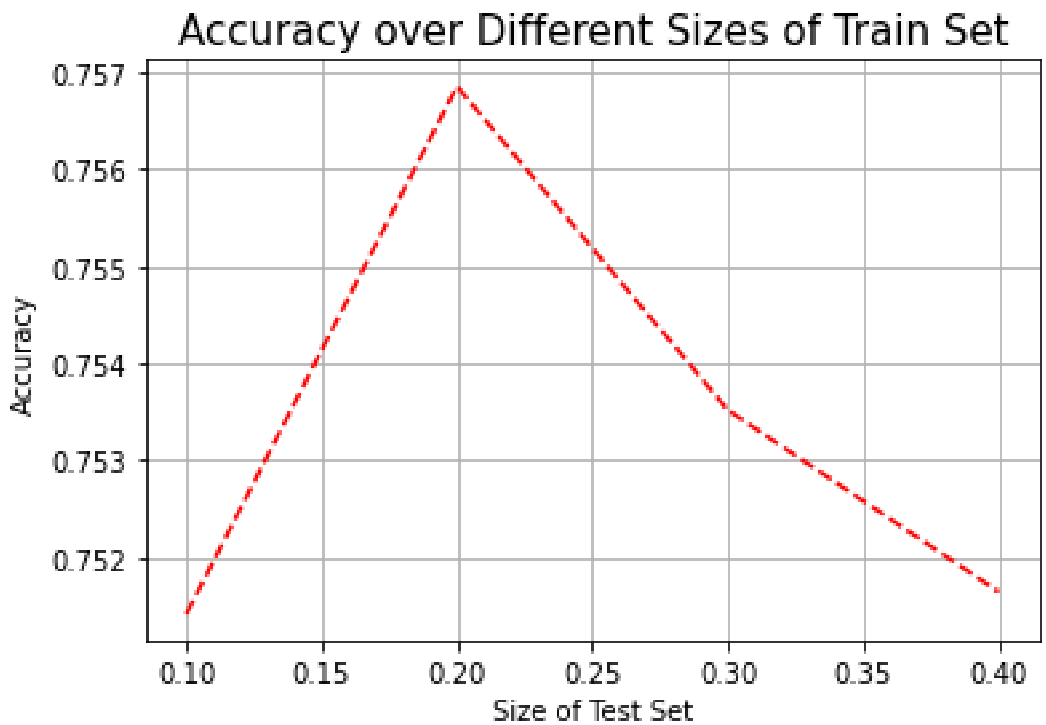
[21]:
```
US_election_data_set_for_supervised
```

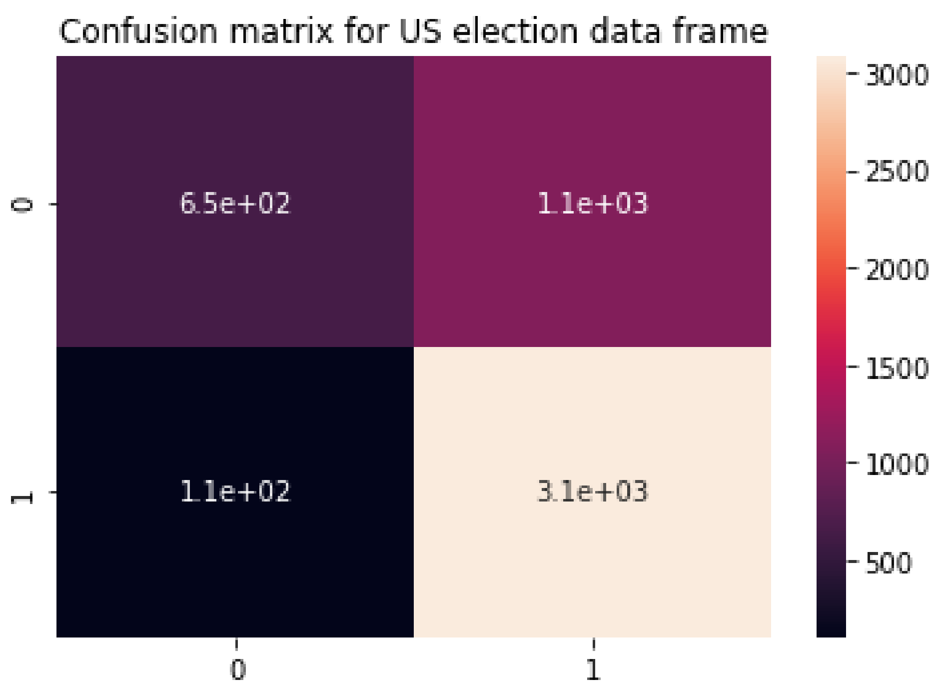
[21]:
```
       state_abbreviation  county  party  candidate  votes  fraction_votes
0                       1     103      0          3    544           0.182
1                       1     103      0          7   2387           0.800
2                       1     114      0          3   2694           0.329
3                       1     114      0          7   5290           0.647
4                       1     127      0          3    222           0.078
…                     …      …      …          …      …             …
24606                  48    2312      1         15      0           0.000
24607                  48    2383      1          6      0           0.000
24608                  48    2383      1          9      0           0.000
24609                  48    2383      1         10      0           0.000
24610                  48    2383      1         15     53           1.000

[24611 rows x 6 columns]
```

[22]:
```python
import numpy as np
from sklearn.naive_bayes import GaussianNB
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, confusion_matrix
import matplotlib.pyplot as plt
import seaborn as sns

def naive_bayes_algorithm(data_frame):
    cols = list(data_frame.columns)
    cols.remove('party')

    X = data_frame[cols].copy()
    y = data_frame['party'].copy()

    def split_test_train(X, y, test_size):
        try:
            return train_test_split(X, y, test_size=test_size, random_state=0)
        except:
           print('Something got wrong - split_test_train')

    def create_naive_bayes_classifier(X, y):
        try:
            model = GaussianNB()
            model.fit(X, y)
            return model
        except:
            print('Something got wrong - create_naive_bayes_classifier')

    accuracy = []
    for ratio in np.arange(0.1, 0.5, 0.1):
```

```
        X_train, X_test, y_train, y_test = split_test_train(X, y, test_size =␣
↪ratio)
        model = create_naive_bayes_classifier(X_train, y_train)
        y_pred = model.predict(X_test)
        accuracy.append(accuracy_score(y_test, y_pred))

    best_accuracy = 0
    x = 0
    split = None
    for i in accuracy:
        x += 1
        if i > best_accuracy:
            best_accuracy = i
            split = "0." + str(x)
    print('The best accuracy is: ' + str(best_accuracy) +'\nThe size of the␣
↪test team is: ' + str(split))

    ratios = np.arange(0.1, 0.5, 0.1)
    plt.grid(True)
    plt.plot(ratios, accuracy, 'r--')
    plt.xlabel('Size of Test Set')
    plt.ylabel('Accuracy')
    plt.title('Accuracy over Different Sizes of Train Set', fontsize=15)
    plt.show()

    X_train, X_test, y_train, y_test = split_test_train(X, y, test_size=0.2)
    model = create_naive_bayes_classifier(X_train, y_train)
    y_pred = model.predict(X_test)
    confusion_matrix(y_test, y_pred)
    sns.heatmap(confusion_matrix(y_test, y_pred), annot=True)
    plt.title('Confusion matrix for US election data frame')
    plt.show()


naive_bayes_algorithm(US_election_data_set_for_supervised)
```

```
The best accuracy is: 0.7568555758683729
The size of the test team is: 0.2
```


Accuracy over Different Sizes of Train Set
Accuracy
0.757
0.756
0.755
0.754
0.753
0.752
0.10
0.15
0.20
0.25
0.30
0.35
0.40
Size of Test Set


Confusion matrix for US election data frame
6.5e+02
1.1e+03
1.1e+02
3.1e+03
0
1
0
1
3000
2500
2000
1500
1000
500

[23]:
```
# The algorithm can predict the chosen party, according to the details of the␣
↪county, above 75 percent! it's a great prediction.
# The second graph, the confusion matrix, shows that the algorithm is right␣
↪most of the time he predicts.
```

[24]:
```
US_election_data_set_for_unsupervised
```

[24]:
```
       votes  fraction_votes  state_abbreviation_AK  state_abbreviation_AL  \
0        544           0.182                      0                      1
1       2387           0.800                      0                      1
2       2694           0.329                      0                      1
3       5290           0.647                      0                      1
4        222           0.078                      0                      1
…          …               …                      …                      …
24606      0           0.000                      0                      0
24607      0           0.000                      0                      0
24608      0           0.000                      0                      0
24609      0           0.000                      0                      0
24610     53           1.000                      0                      0

       state_abbreviation_AR  state_abbreviation_AZ  state_abbreviation_CA  \
0                          0                      0                      0
1                          0                      0                      0
2                          0                      0                      0
3                          0                      0                      0
4                          0                      0                      0
…                          …                      …                      …
24606                      0                      0                      0
24607                      0                      0                      0
24608                      0                      0                      0
24609                      0                      0                      0
24610                      0                      0                      0

       state_abbreviation_CO  state_abbreviation_CT  state_abbreviation_DE  \
0                          0                      0                      0
1                          0                      0                      0
2                          0                      0                      0
3                          0                      0                      0
4                          0                      0                      0
…                          …                      …                      …
24606                      0                      0                      0
24607                      0                      0                      0
24608                      0                      0                      0
24609                      0                      0                      0
24610                      0                      0                      0

       …  candidate_Donald Trump  candidate_Hillary Clinton  \
```

```
0      …                      0                           0
1      …                      0                           1
2      …                      0                           0
3      …                      0                           1
4      …                      0                           0
…    …                    …                          …
24606  …                      0                           0
24607  …                      1                           0
24608  …                      0                           0
24609  …                      0                           0
24610  …                      0                           0

       candidate_Jeb Bush  candidate_John Kasich  candidate_Marco Rubio  \
0                       0                      0                      0
1                       0                      0                      0
2                       0                      0                      0
3                       0                      0                      0
4                       0                      0                      0
…                   …                      …                      …
24606                   0                      0                      0
24607                   0                      0                      0
24608                   0                      1                      0
24609                   0                      0                      1
24610                   0                      0                      0

       candidate_Martin O'Malley  candidate_Mike Huckabee  \
0                              0                        0
1                              0                        0
2                              0                        0
3                              0                        0
4                              0                        0
…                          …                        …
24606                          0                        0
24607                          0                        0
24608                          0                        0
24609                          0                        0
24610                          0                        0

       candidate_Rand Paul  candidate_Rick Santorum  candidate_Ted Cruz
0                        0                        0                  0
1                        0                        0                  0
2                        0                        0                  0
3                        0                        0                  0
4                        0                        0                  0
…                    …                        …                  …
24606                    0                        0                  1
24607                    0                        0                  0
```

```
24608                          0                          0                     0
24609                          0                          0                     0
24610                          0                          0                     1

[24611 rows x 2702 columns]
```

[34]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
from sklearn.model_selection import train_test_split

X = US_election_data_set_for_supervised.drop('party',axis=1)
y = US_election_data_set_for_supervised['party']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.30)
```

[35]:
```
from sklearn.tree import DecisionTreeClassifier

dtree = DecisionTreeClassifier()
dtree.fit(X_train,y_train)
```

[35]:
```
DecisionTreeClassifier()
```

[36]:
```
predictions = dtree.predict(X_test)
```

[37]:
```
from sklearn.metrics import classification_report,confusion_matrix
```

[38]:
```
print(classification_report(y_test,predictions))
```

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00      2701
           1       1.00      1.00      1.00      4683

    accuracy                           1.00      7384
   macro avg       1.00      1.00      1.00      7384
weighted avg       1.00      1.00      1.00      7384
```

[39]:
```
print(confusion_matrix(y_test,predictions))
```

```
[[2698    3]
 [   3 4680]]
```

[31]:
```
# These measure is coming together with the conclusions above and stronger the
↪prediction of the party.
# The algorithm is right 7376 times from 7384 cases! The algorithm can predict
↪the party absolutely.
```

[47]:
```
from IPython.display import Image
from sklearn.tree import export_graphviz
import pydot

features = list(US_election_data_set_for_supervised.columns[1:])
features
```

[47]:
```
['county', 'party', 'candidate', 'votes', 'fraction_votes']
```