In [1]:
```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.metrics import accuracy_score
from sklearn.metrics import f1_score,recall_score,confusion_matrix,precision_scor
from sklearn.metrics import confusion_matrix
from sklearn.metrics import ConfusionMatrixDisplay
from sklearn.preprocessing import LabelEncoder
```

In [2]:
```python
df = pd.read_csv('census_labeled (2).csv')
df
```

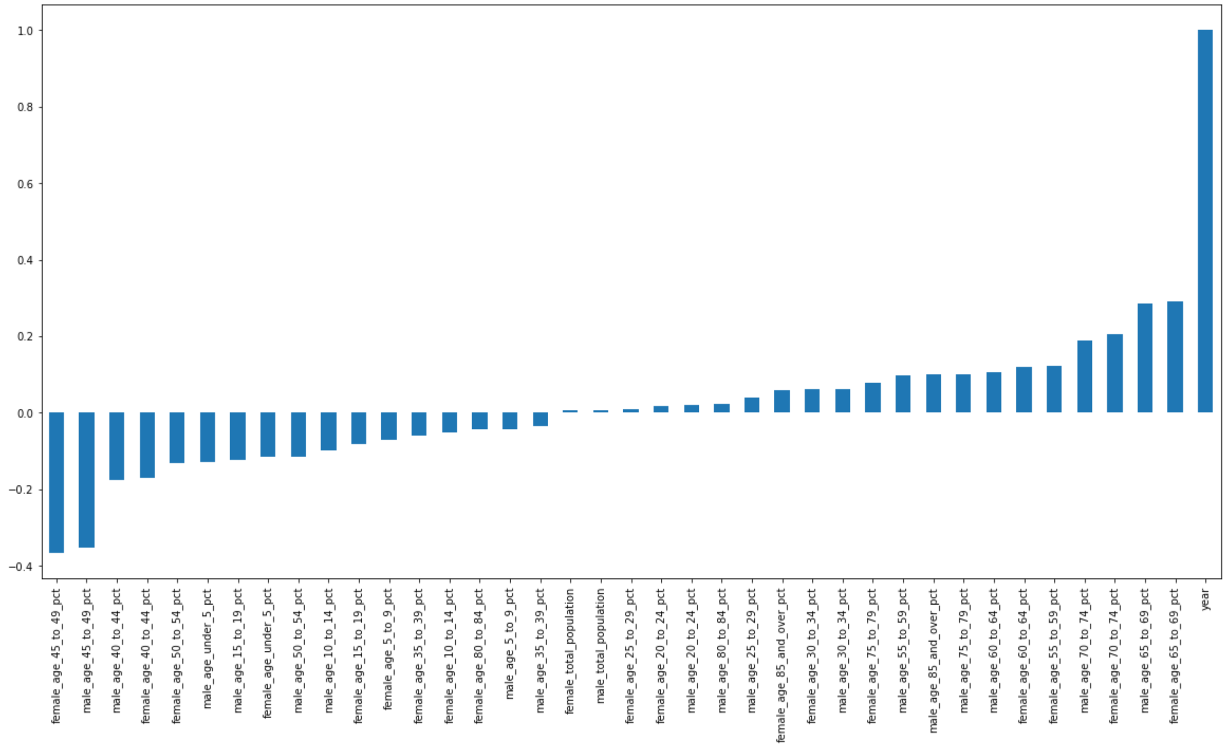
Out[2]:

	county name	state	county_population_increased_2015_2016	year	female_total_population	fe
0	Stark County	Ohio	False	2010	192651	
1	Summit County	Ohio	False	2010	279592	
2	Trumbull County	Ohio	False	2010	108490	
3	Tuscarawas County	Ohio	False	2010	47279	
4	Warren County	Ohio	True	2010	105706	
...	...	...	...	...	...	
4946	Toa Alta Municipio	Puerto Rico	False	2015	38559	
4947	Toa Baja Municipio	Puerto Rico	False	2015	43530	
4948	Trujillo Alto Municipio	Puerto Rico	False	2015	36804	
4949	Bayamón Municipio	Puerto Rico	False	2015	99486	
4950	Mayagüez Municipio	Puerto Rico	False	2015	41540	

4951 rows × 42 columns

In [3]:
```
df.corr().iloc[:,0].sort_values().plot(kind='bar',figsize=(20,10))
```

Out[3]: <AxesSubplot:>



In [4]:
```
df.female_age_under_5_pct = df.female_total_population*df.female_age_under_5_pct/
df.female_age_5_to_9_pct = df.female_total_population*df.female_age_5_to_9_pct/10
df.female_age_10_to_14_pct = df.female_total_population*df.female_age_10_to_14_pc
df.female_age_15_to_19_pct = df.female_total_population*df.female_age_15_to_19_pc
df.female_age_20_to_24_pct = df.female_total_population*df.female_age_20_to_24_pc
df.female_age_25_to_29_pct = df.female_total_population*df.female_age_25_to_29_pc
df.female_age_30_to_34_pct = df.female_total_population*df.female_age_30_to_34_pc
df.female_age_35_to_39_pct = df.female_total_population*df.female_age_35_to_39_pc
df.female_age_40_to_44_pct = df.female_total_population*df.female_age_40_to_44_pc
df.female_age_45_to_49_pct = df.female_total_population*df.female_age_45_to_49_pc
df.female_age_50_to_54_pct = df.female_total_population*df.female_age_50_to_54_pc
df.female_age_55_to_59_pct = df.female_total_population*df.female_age_55_to_59_pc
df.female_age_60_to_64_pct = df.female_total_population*df.female_age_60_to_64_pc
df.female_age_65_to_69_pct = df.female_total_population*df.female_age_65_to_69_pc
df.female_age_70_to_74_pct = df.female_total_population*df.female_age_70_to_74_pc
df.female_age_75_to_79_pct = df.female_total_population*df.female_age_75_to_79_pc
df.female_age_80_to_84_pct = df.female_total_population*df.female_age_80_to_84_pc
df.female_age_85_and_over_pct = df.female_total_population*df.female_age_85_and_c
```

In [5]:
```
df.male_age_under_5_pct = df.male_total_population*df.male_age_under_5_pct/100
df.male_age_5_to_9_pct = df.male_total_population*df.male_age_5_to_9_pct/100
df.male_age_10_to_14_pct = df.male_total_population*df.male_age_10_to_14_pct/100
df.male_age_15_to_19_pct = df.male_total_population*df.male_age_15_to_19_pct/100
df.male_age_20_to_24_pct = df.male_total_population*df.male_age_20_to_24_pct/100
df.male_age_25_to_29_pct = df.male_total_population*df.male_age_25_to_29_pct/100
df.male_age_30_to_34_pct = df.male_total_population*df.male_age_30_to_34_pct/100
df.male_age_35_to_39_pct = df.male_total_population*df.male_age_35_to_39_pct/100
df.male_age_40_to_44_pct = df.male_total_population*df.male_age_40_to_44_pct/100
df.male_age_45_to_49_pct = df.male_total_population*df.male_age_45_to_49_pct/100
df.male_age_50_to_54_pct = df.male_total_population*df.male_age_50_to_54_pct/100
df.male_age_55_to_59_pct = df.male_total_population*df.male_age_55_to_59_pct/100
df.male_age_60_to_64_pct = df.male_total_population*df.male_age_60_to_64_pct/100
df.male_age_65_to_69_pct = df.male_total_population*df.male_age_65_to_69_pct/100
df.male_age_70_to_74_pct = df.male_total_population*df.male_age_70_to_74_pct/100
df.male_age_75_to_79_pct = df.male_total_population*df.male_age_75_to_79_pct/100
df.male_age_80_to_84_pct = df.male_total_population*df.male_age_80_to_84_pct/100
df.male_age_85_and_over_pct = df.male_total_population*df.male_age_85_and_over_pc
```

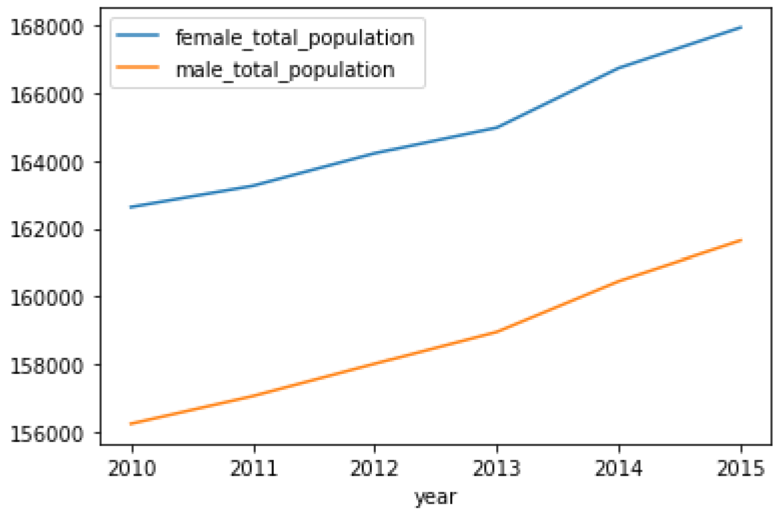
In [6]:
```
df
```

Out[6]:

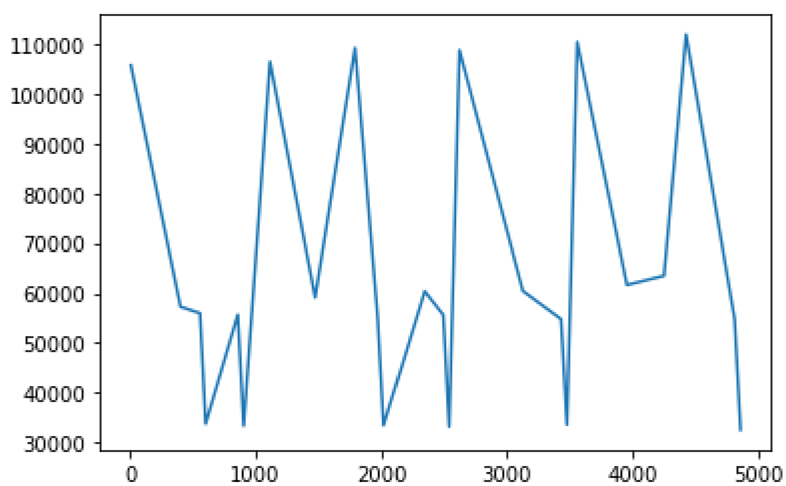
	county name	state	county_population_increased_2015_2016	year	female_total_population	fe
0	Stark County	Ohio	False	2010	192651	
1	Summit County	Ohio	False	2010	279592	
2	Trumbull County	Ohio	False	2010	108490	
3	Tuscarawas County	Ohio	False	2010	47279	
4	Warren County	Ohio	True	2010	105706	
...	...	...	...	...	...	
4946	Toa Alta Municipio	Puerto Rico	False	2015	38559	
4947	Toa Baja Municipio	Puerto Rico	False	2015	43530	
4948	Trujillo Alto Municipio	Puerto Rico	False	2015	36804	
4949	Bayamón Municipio	Puerto Rico	False	2015	99486	
4950	Mayagüez Municipio	Puerto Rico	False	2015	41540	

4951 rows × 42 columns

In [7]:
```python
df.groupby('year').mean()[['female_total_population' , 'male_total_population']].
```
Out[7]: <AxesSubplot:xlabel='year'>

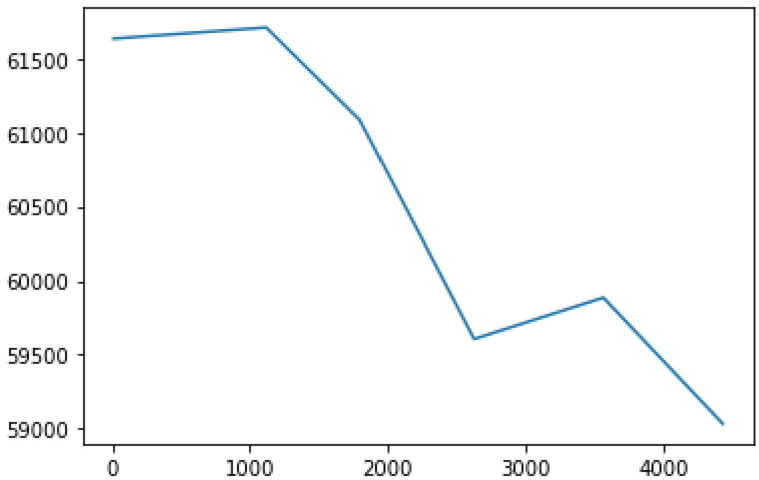


In [8]:
```python
df[df['county name'] == 'Warren County']['female_total_population'].plot()
```
Out[8]: <AxesSubplot:>



In [9]:
```
df[df['county name'] == 'Comanche County']['female_total_population'].plot()
```

Out[9]: <AxesSubplot:>



In [10]:
```
df[df['year'] == 2015].groupby('state').mean()
```

Out[10]:

state	year	female_total_population	female_age_under_5_pct	female_age_5_to_9_pct	fe
Alabama	2015.0	89143.095238	5050.608714	5468.212810	
Alaska	2015.0	79403.333333	6072.043000	5882.711333	
Arizona	2015.0	335736.100000	20486.708400	22051.816000	
Arkansas	2015.0	77046.181818	5067.847182	5069.264545	
California	2015.0	486521.025000	30325.205850	30915.998200	
Colorado	2015.0	195333.500000	11878.393667	12676.465250	
Connecticut	2015.0	229852.875000	11458.595875	12754.876875	
Delaware	2015.0	162575.333333	9013.996000	8918.158000	
District of Columbia	2015.0	352523.000000	21151.380000	16216.058000	
Florida	2015.0	251238.200000	12997.283000	13441.668250	
Georgia	2015.0	110586.833333	6840.602417	7266.327694	
Hawaii	2015.0	176723.500000	11341.124000	10514.779500	
Idaho	2015.0	89632.166667	5951.454167	6342.694333	
Illinois	2015.0	245176.521739	14505.460957	15026.626130	
Indiana	2015.0	97037.800000	6028.551080	6228.381920	
Iowa	2015.0	81558.600000	5211.838300	5106.576100	
Kansas	2015.0	111713.250000	7606.679625	7704.536375	
Kentucky	2015.0	84930.461538	5253.495846	5179.018615	
Louisiana	2015.0	102766.941176	6371.775824	6699.221059	
Maine	2015.0	80153.333333	3859.830167	3872.571333	
Maryland	2015.0	184531.625000	10823.508938	10920.819187	
Massachusetts	2015.0	290250.166667	14846.490917	15489.207750	
Michigan	2015.0	149194.172414	8333.588483	8640.492931	
Minnesota	2015.0	139698.285714	8812.171357	8766.692857	
Mississippi	2015.0	68622.300000	4423.480000	4564.694600	
Missouri	2015.0	124968.764706	7368.695706	7840.588706	
Montana	2015.0	51600.333333	3022.145833	3310.868167	
Nebraska	2015.0	173302.666667	12425.769333	12289.589000	
Nevada	2015.0	640444.000000	38844.771000	40040.997500	
New Hampshire	2015.0	96854.500000	4476.466667	5137.968667	
New Jersey	2015.0	218243.857143	12278.370238	12774.242524	
New Mexico	2015.0	83724.800000	5295.618900	5459.034100	

	year	female_total_population	female_age_under_5_pct	female_age_5_to_9_pct	fe
state					
New York	2015.0	248220.717949	14191.504615	13514.337385	
North Carolina	2015.0	103859.600000	6086.529000	6521.071775	
North Dakota	2015.0	50070.750000	3557.710500	3227.447500	
Ohio	2015.0	128012.333333	7258.961256	7438.421436	
Oklahoma	2015.0	110964.454545	7368.013000	7331.494909	
Oregon	2015.0	120590.733333	6727.166333	7347.152867	
Pennsylvania	2015.0	151494.425000	8135.077100	8534.167850	
Puerto Rico	2015.0	71194.636364	3195.394364	3553.919091	
Rhode Island	2015.0	129885.250000	6411.557250	7384.320500	
South Carolina	2015.0	101512.904762	5775.679286	6401.324095	
South Dakota	2015.0	72635.000000	5099.173000	4320.046000	
Tennessee	2015.0	116562.950000	7025.443650	7287.258700	
Texas	2015.0	228488.509434	16209.197415	16762.777283	
Utah	2015.0	210931.500000	17487.250500	17747.376500	
Vermont	2015.0	82838.000000	4307.576000	3727.710000	
Virginia	2015.0	102539.366667	6344.966900	6411.055667	
Washington	2015.0	175196.000000	10762.641263	11107.512947	
West Virginia	2015.0	53409.142857	2967.646571	2655.256000	
Wisconsin	2015.0	96752.391304	5626.738174	6115.312783	
Wyoming	2015.0	44979.000000	3015.035500	2896.641500	

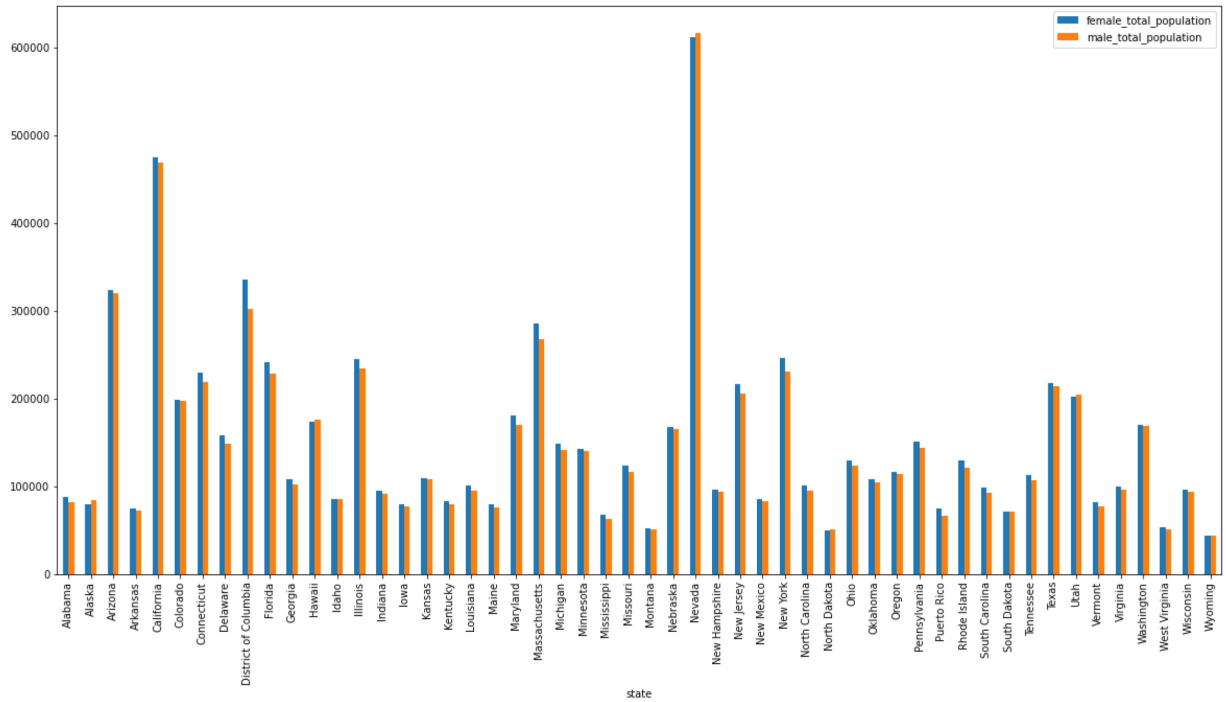
52 rows × 39 columns

In [11]:
```
df.nunique()
```

Out[11]:
```
county name                                 662
state                                        52
county_population_increased_2015_2016         2
year                                          6
female_total_population                    4859
female_age_under_5_pct                     4945
female_age_5_to_9_pct                      4945
female_age_10_to_14_pct                    4951
female_age_15_to_19_pct                    4945
female_age_20_to_24_pct                    4948
female_age_25_to_29_pct                    4949
female_age_30_to_34_pct                    4946
female_age_35_to_39_pct                    4951
female_age_40_to_44_pct                    4948
female_age_45_to_49_pct                    4947
female_age_50_to_54_pct                    4946
female_age_55_to_59_pct                    4946
female_age_60_to_64_pct                    4947
female_age_65_to_69_pct                    4948
female_age_70_to_74_pct                    4945
female_age_75_to_79_pct                    4943
female_age_80_to_84_pct                    4937
female_age_85_and_over_pct                 4943
male_total_population                      4862
male_age_under_5_pct                       4948
male_age_5_to_9_pct                        4947
male_age_10_to_14_pct                      4945
male_age_15_to_19_pct                      4945
male_age_20_to_24_pct                      4944
male_age_25_to_29_pct                      4949
male_age_30_to_34_pct                      4949
male_age_35_to_39_pct                      4947
male_age_40_to_44_pct                      4946
male_age_45_to_49_pct                      4946
male_age_50_to_54_pct                      4947
male_age_55_to_59_pct                      4949
male_age_60_to_64_pct                      4946
male_age_65_to_69_pct                      4945
male_age_70_to_74_pct                      4942
male_age_75_to_79_pct                      4943
male_age_80_to_84_pct                      4943
male_age_85_and_over_pct                   4933
dtype: int64
```

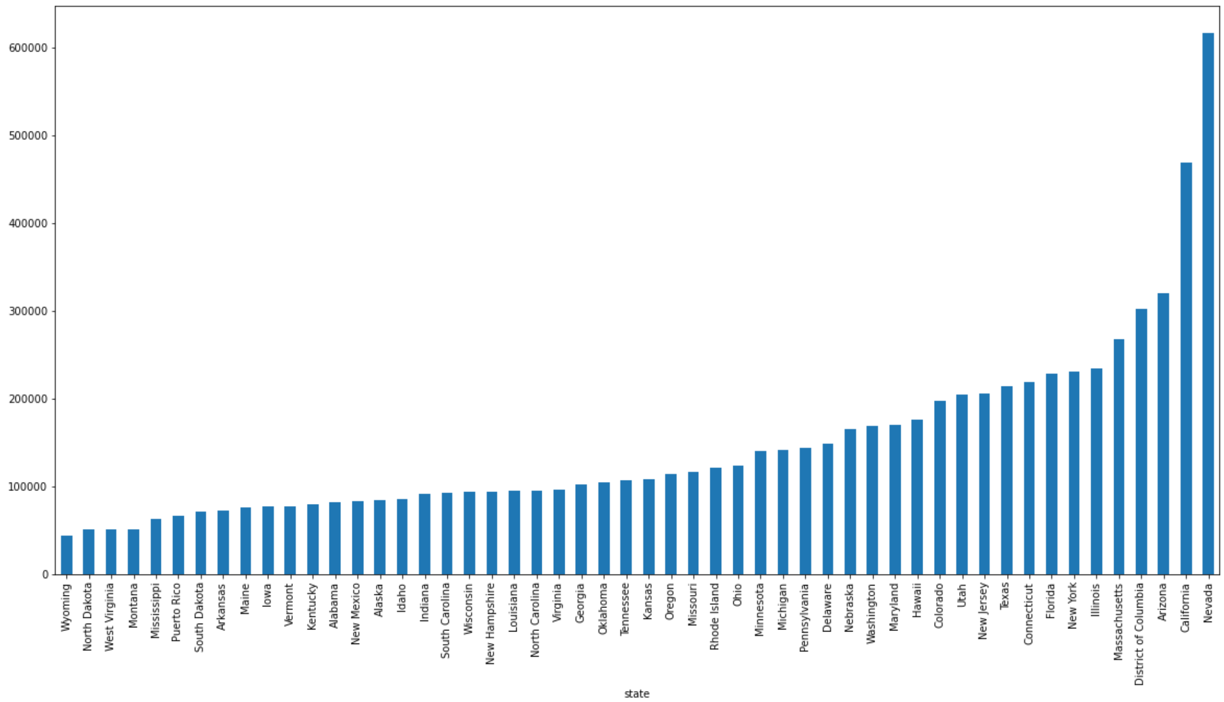
In [12]:
```
df.groupby('state').mean()[['female_total_population','male_total_population']].p
```

Out[12]: <AxesSubplot:xlabel='state'>



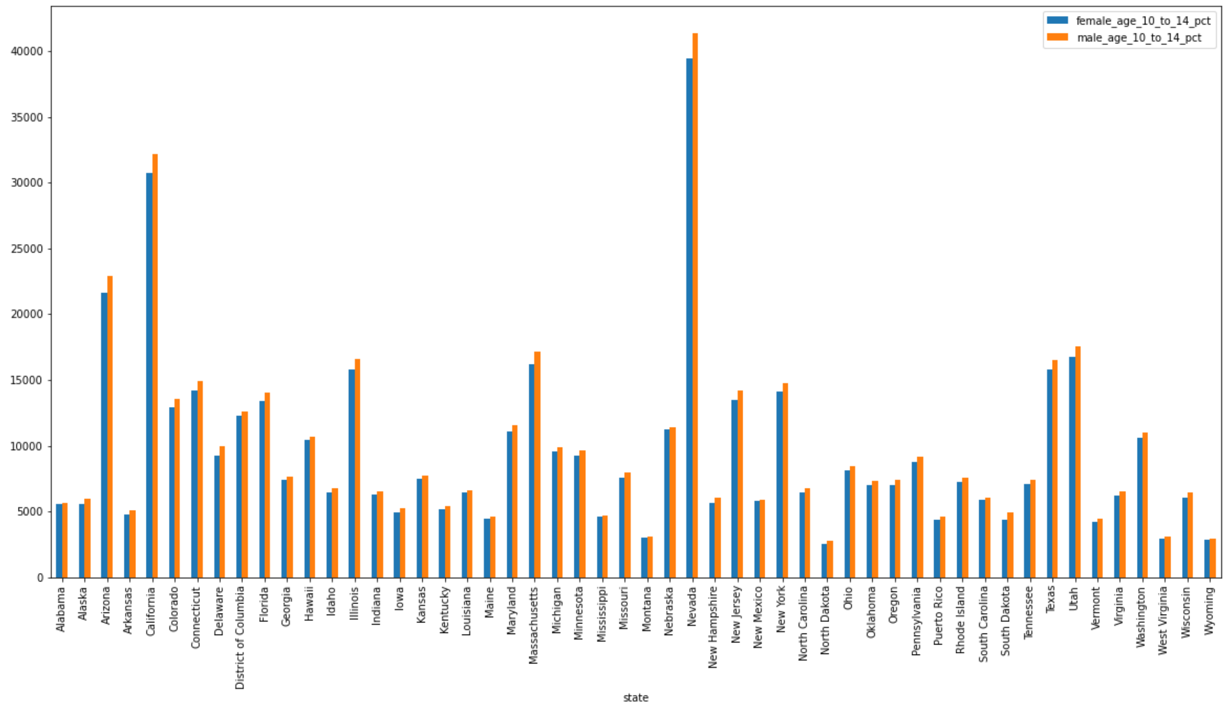
In [13]:
```
df.groupby('state').mean()['male_total_population'].sort_values().plot(kind = 'ba
```

Out[13]: <AxesSubplot:xlabel='state'>



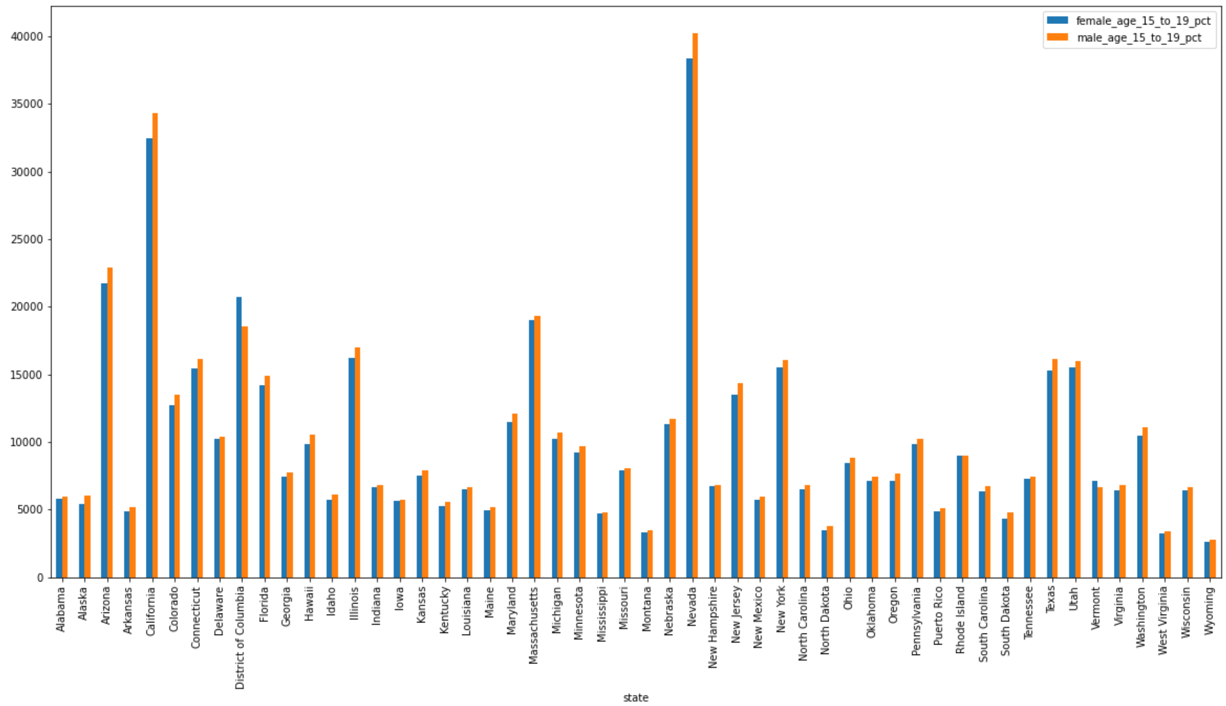
In [14]:
```
df.groupby('state').mean()[['female_age_10_to_14_pct','male_age_10_to_14_pct']].p
```

Out[14]: <AxesSubplot:xlabel='state'>



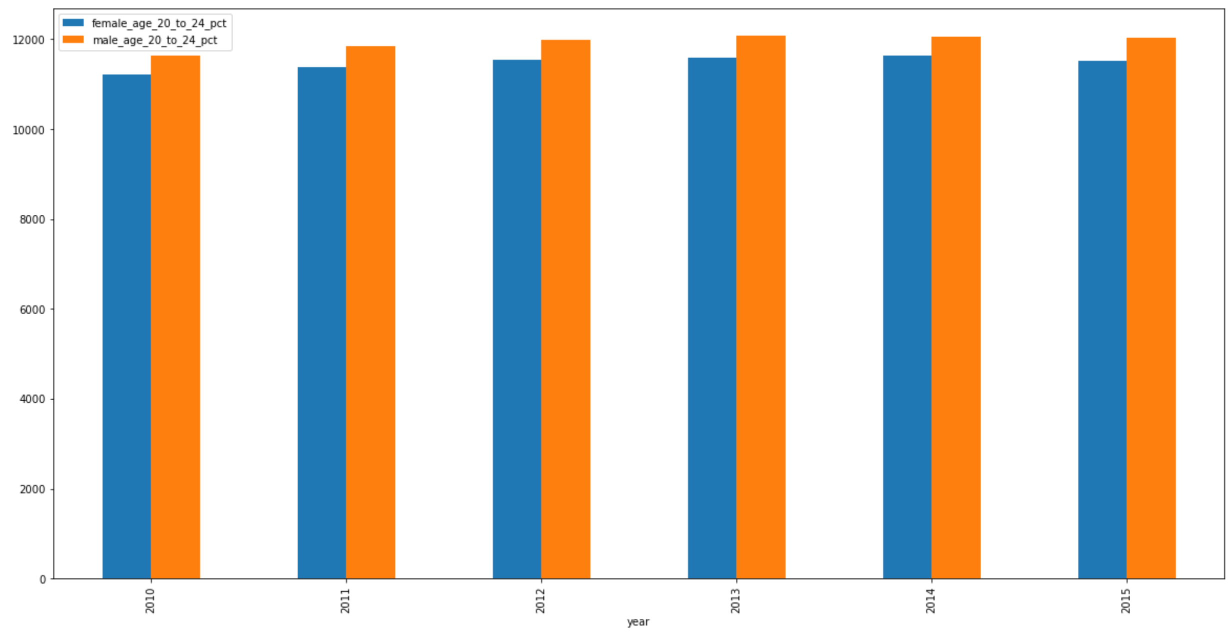
In [15]:
```
df.groupby('state').mean()[['female_age_15_to_19_pct','male_age_15_to_19_pct']].p
```

Out[15]: <AxesSubplot:xlabel='state'>



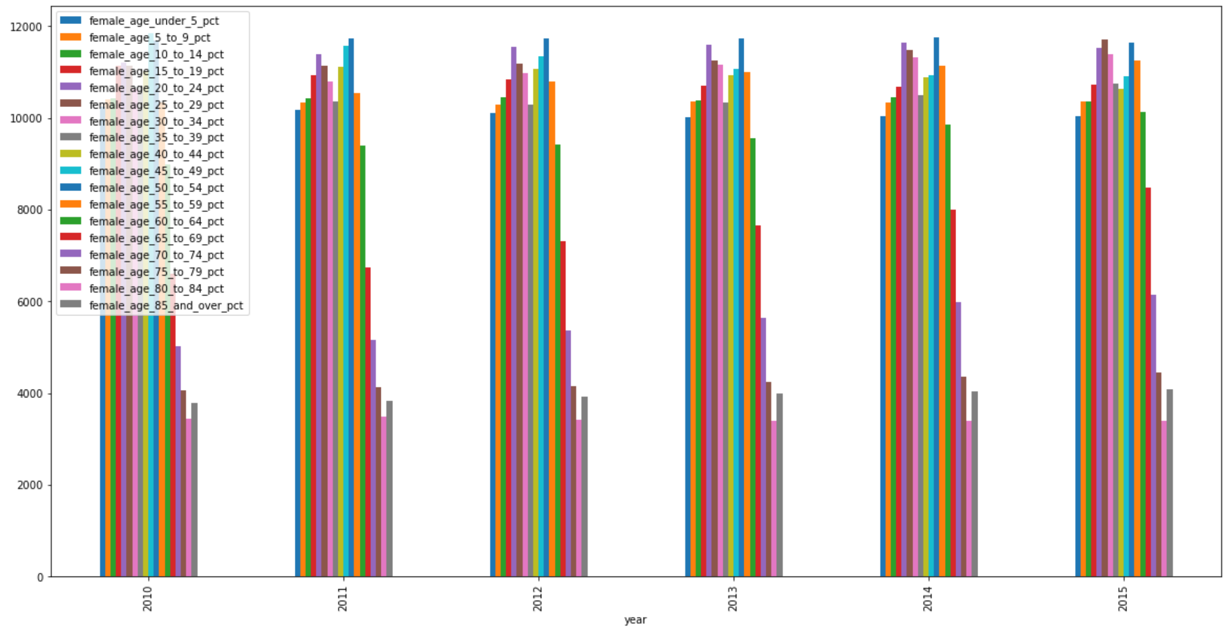
In [16]:
```
df.groupby('year').mean()[['female_age_20_to_24_pct','male_age_20_to_24_pct']].pl
```

Out[16]: <AxesSubplot:xlabel='year'>



In [17]:
```
df.groupby('year').mean()[['female_age_under_5_pct','female_age_5_to_9_pct','fema
```

Out[17]: <AxesSubplot:xlabel='year'>



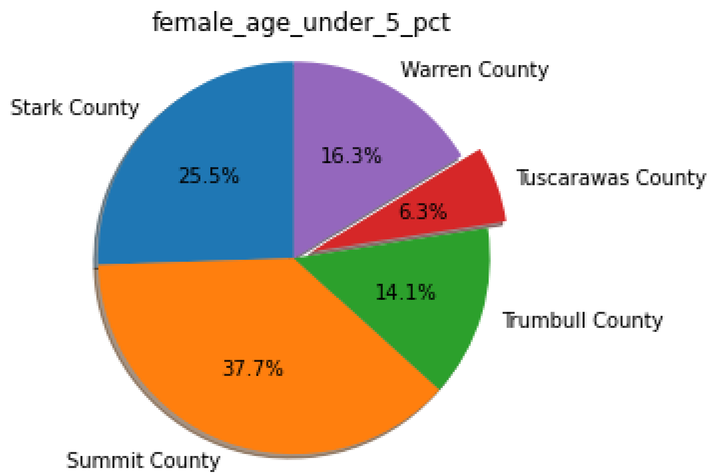
In [18]:
```
df
```

Out[18]:

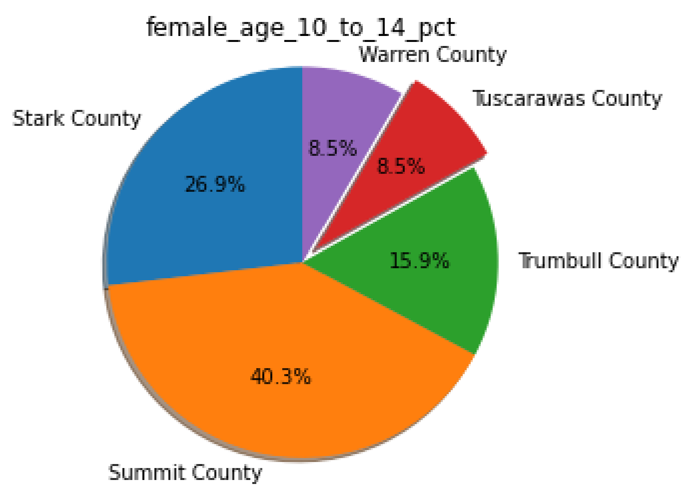
	county name	state	county_population_increased_2015_2016	year	female_total_population	fe
0	Stark County	Ohio	False	2010	192651	
1	Summit County	Ohio	False	2010	279592	
2	Trumbull County	Ohio	False	2010	108490	
3	Tuscarawas County	Ohio	False	2010	47279	
4	Warren County	Ohio	True	2010	105706	
...	...	...	...	...	...	
4946	Toa Alta Municipio	Puerto Rico	False	2015	38559	
4947	Toa Baja Municipio	Puerto Rico	False	2015	43530	
4948	Trujillo Alto Municipio	Puerto Rico	False	2015	36804	
4949	Bayamón Municipio	Puerto Rico	False	2015	99486	
4950	Mayagüez Municipio	Puerto Rico	False	2015	41540	

4951 rows × 42 columns

In [19]:
```
import matplotlib.pyplot as plt

df1 = [10595.805, 15657.152, 5858.460, 2600.345, 6765.184,]
label = ['Stark County', 'Summit County', 'Trumbull County', 'Tuscarawas County',
plt.pie(df1, labels = label, autopct = '%1.1f%%', explode = [0,0,0,0.1,0], shadow
plt.title('female_age_under_5_pct')
plt.axis('equal')
plt.show()
```



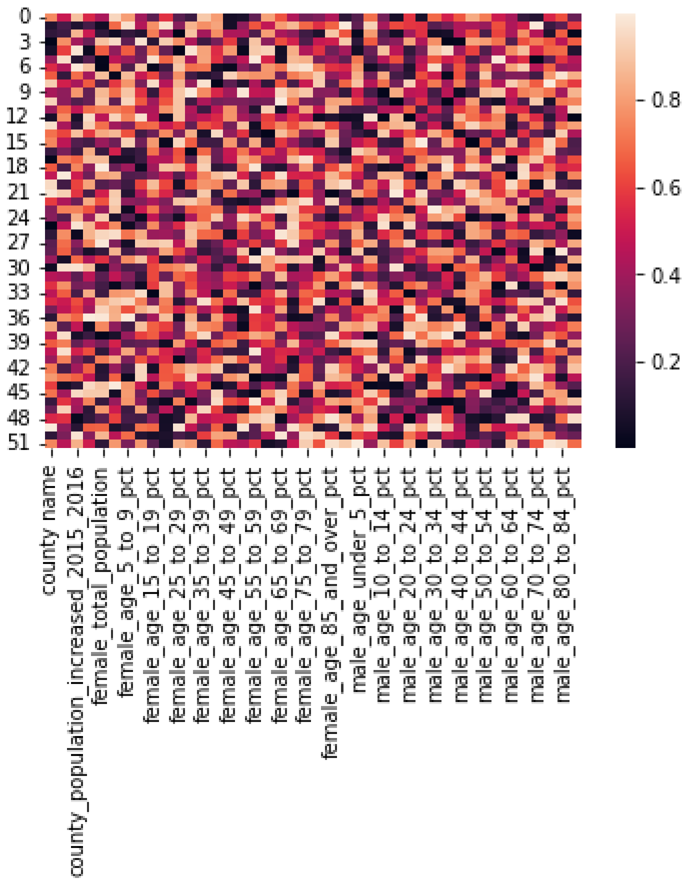
In [20]:
```
import matplotlib.pyplot as plt

df1 = [11559.060 , 17334.704 , 6834.870 , 3640.483 , 3640.483 ,]
label = ['Stark County', 'Summit County', 'Trumbull County', 'Tuscarawas County',

plt.pie(df1, labels = label, autopct = '%1.1f%%', explode = [0,0,0,0.1,0], shadow
plt.title('female_age_10_to_14_pct')
plt.axis('equal')
plt.show()
```



```
In [21]: print(df[['county name', 'state', 'county_population_increased_2015_2016', 'year'
       'female_total_population', 'female_age_under_5_pct',
       'female_age_5_to_9_pct', 'female_age_10_to_14_pct',
       'female_age_15_to_19_pct', 'female_age_20_to_24_pct',
       'female_age_25_to_29_pct', 'female_age_30_to_34_pct',
       'female_age_35_to_39_pct', 'female_age_40_to_44_pct',
       'female_age_45_to_49_pct', 'female_age_50_to_54_pct',
       'female_age_55_to_59_pct', 'female_age_60_to_64_pct',
       'female_age_65_to_69_pct', 'female_age_70_to_74_pct',
       'female_age_75_to_79_pct', 'female_age_80_to_84_pct',
       'female_age_85_and_over_pct', 'male_total_population',
       'male_age_under_5_pct', 'male_age_5_to_9_pct', 'male_age_10_to_14_pct',
       'male_age_15_to_19_pct', 'male_age_20_to_24_pct',
       'male_age_25_to_29_pct', 'male_age_30_to_34_pct',
       'male_age_35_to_39_pct', 'male_age_40_to_44_pct',
       'male_age_45_to_49_pct', 'male_age_50_to_54_pct',
       'male_age_55_to_59_pct', 'male_age_60_to_64_pct',
       'male_age_65_to_69_pct', 'male_age_70_to_74_pct',
       'male_age_75_to_79_pct', 'male_age_80_to_84_pct',
       'male_age_85_and_over_pct']].corr(method = 'pearson'))
```

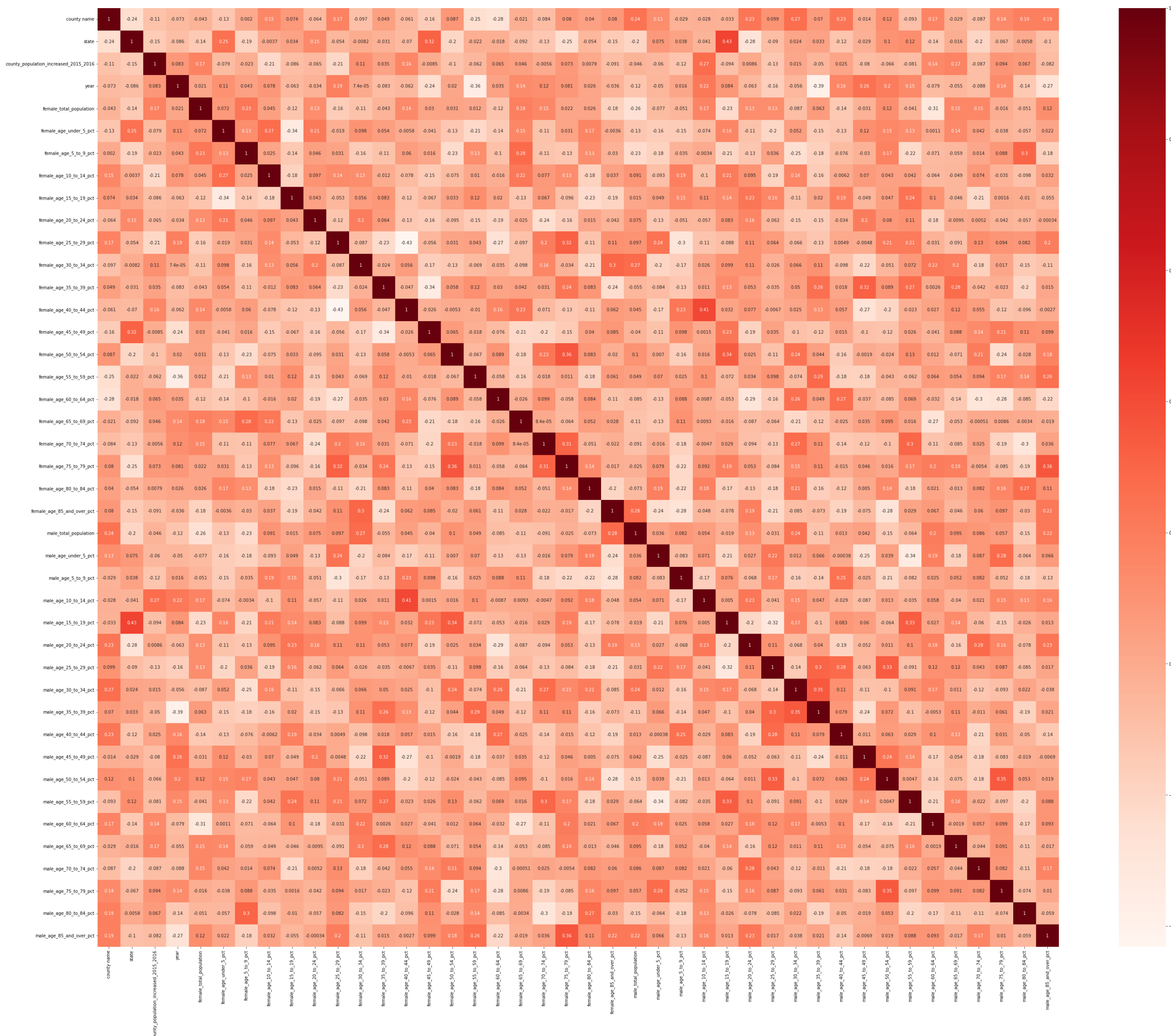
```
                                year  female_total_population  \
year                        1.000000                 0.006365
female_total_population     0.006365                 1.000000
female_age_under_5_pct     -0.003691                 0.991554
female_age_5_to_9_pct      -0.000372                 0.989353
female_age_10_to_14_pct    -0.000764                 0.990788
female_age_15_to_19_pct    -0.007319                 0.993002
female_age_20_to_24_pct     0.005537                 0.988849
female_age_25_to_29_pct     0.008643                 0.988602
female_age_30_to_34_pct     0.012536                 0.993259
female_age_35_to_39_pct     0.001890                 0.995898
female_age_40_to_44_pct    -0.006627                 0.996320
female_age_45_to_49_pct    -0.016901                 0.996235
female_age_50_to_54_pct    -0.000530                 0.996743
female_age_55_to_59_pct     0.017757                 0.995282
female_age_60_to_64_pct     0.022927                 0.993040
female_age_65_to_69_pct     0.054650                 0.983539
female_age_70_to_74_pct     0.044979                 0.977261
female_age_75_to_79_pct     0.019371                 0.973695
```

In [22]:
```python
dd = pd.DataFrame(np.random.random((52,42)), columns = ['county name', 'state', '
        'female_total_population', 'female_age_under_5_pct',
        'female_age_5_to_9_pct', 'female_age_10_to_14_pct',
        'female_age_15_to_19_pct', 'female_age_20_to_24_pct',
        'female_age_25_to_29_pct', 'female_age_30_to_34_pct',
        'female_age_35_to_39_pct', 'female_age_40_to_44_pct',
        'female_age_45_to_49_pct', 'female_age_50_to_54_pct',
        'female_age_55_to_59_pct', 'female_age_60_to_64_pct',
        'female_age_65_to_69_pct', 'female_age_70_to_74_pct',
        'female_age_75_to_79_pct', 'female_age_80_to_84_pct',
        'female_age_85_and_over_pct', 'male_total_population',
        'male_age_under_5_pct', 'male_age_5_to_9_pct', 'male_age_10_to_14_pct',
        'male_age_15_to_19_pct', 'male_age_20_to_24_pct',
        'male_age_25_to_29_pct', 'male_age_30_to_34_pct',
        'male_age_35_to_39_pct', 'male_age_40_to_44_pct',
        'male_age_45_to_49_pct', 'male_age_50_to_54_pct',
        'male_age_55_to_59_pct', 'male_age_60_to_64_pct',
        'male_age_65_to_69_pct', 'male_age_70_to_74_pct',
        'male_age_75_to_79_pct', 'male_age_80_to_84_pct',
        'male_age_85_and_over_pct'])
p = sns.heatmap(dd)
```



In [23]:
```python
cor = dd.corr()
```

In [24]:
```python
import matplotlib.pyplot as plt
import seaborn as sns

plt.figure(figsize=(50,40))

sns.heatmap(cor, annot=True , cmap = plt.cm.Reds)
plt.show()
```



In [25]:
```
df['county name'].unique()
```

Out[25]:
```
array(['Stark County', 'Summit County', 'Trumbull County',
       'Tuscarawas County', 'Warren County', 'Wayne County',
       'Wood County', 'Canadian County', 'Cleveland County',
       'Comanche County', 'Creek County', 'Muskogee County',
       'Oklahoma County', 'Payne County', 'Pottawatomie County',
       'Rogers County', 'Tulsa County', 'Wagoner County', 'Benton County',
       'Clackamas County', 'Deschutes County', 'Douglas County',
       'Jackson County', 'Josephine County', 'Klamath County',
       'Lane County', 'Linn County', 'Marion County', 'Multnomah County',
       'Polk County', 'Umatilla County', 'Washington County',
       'Yamhill County', 'Adams County', 'Allegheny County',
       'Armstrong County', 'Beaver County', 'Berks County',
       'Blair County', 'Bucks County', 'Butler County', 'Cambria County',
       'Carbon County', 'Centre County', 'Chester County',
       'Clearfield County', 'Columbia County', 'Crawford County',
       'Cumberland County', 'Dauphin County', 'Delaware County',
       'Erie County', 'Fayette County', 'Franklin County',
       'Indiana County', 'Lackawanna County', 'Lancaster County',
       'Lawrence County', 'Lebanon County', 'Lehigh County',
```

In [26]:
```
df['county name']=df['county name'].astype('category')
df['county name']=df['county name'].cat.codes
df['state']=df['state'].astype('category')
df['state']=df['state'].cat.codes
df['county_population_increased_2015_2016']=df['county_population_increased_2015_
df['county_population_increased_2015_2016']=df['county_population_increased_2015_
```

In [27]:
```
# le = LabelEncoder()
# encoded = le.fit_transform(df[['state']])
# encoded
```

In [28]:
```
# le.inverse_transform(encoded)
```

In [29]:
```
# le = LabelEncoder()
# encoded_1 = le.fit_transform(df[['county_population_increased_2015_2016']])
# encoded_1
```

In [30]:
```
# le.inverse_transform(encoded_1)
```

In [31]:
```
# le = LabelEncoder()
# encoded_2 = le.fit_transform(df[['county name']])
# encoded_2
```

In [ ]:

In [32]:
```
df['county_population_increased_2015_2016'].unique()
```

Out[32]:
```
array([ 0,  1, -1], dtype=int8)
```

In [33]:
```
df
```

Out[33]:

	county name	state	county_population_increased_2015_2016	year	female_total_population	female_
0	572	35	0	2010	192651	
1	580	35	0	2010	279592	
2	602	35	0	2010	108490	
3	606	35	0	2010	47279	
4	627	35	1	2010	105706	
...	...	...	...	...	...	
4946	594	39	0	2015	38559	
4947	595	39	0	2015	43530	
4948	601	39	0	2015	36804	
4949	40	39	0	2015	99486	
4950	371	39	0	2015	41540	

4951 rows × 42 columns

In [34]:
```
val = []
ind = []
for i,x in enumerate(df.county_population_increased_2015_2016):
    if x==-1:
        val.append(x)
        ind.append(i)
print(ind)
```

```
[398, 1472, 2342, 3124, 3954, 4250]
```

In [35]:
```
df.drop(index = ind , inplace = True)
```

In [36]:
```
df
```

Out[36]:

	county name	state	county_population_increased_2015_2016	year	female_total_population	female_
0	572	35	0	2010	192651	
1	580	35	0	2010	279592	
2	602	35	0	2010	108490	
3	606	35	0	2010	47279	
4	627	35	1	2010	105706	
...	...	...	...	...	...	
4946	594	39	0	2015	38559	
4947	595	39	0	2015	43530	
4948	601	39	0	2015	36804	
4949	40	39	0	2015	99486	
4950	371	39	0	2015	41540	

4945 rows × 42 columns

In [37]:
```
X =df[['county name', 'state', 'year',
       'female_total_population', 'female_age_under_5_pct',
       'female_age_5_to_9_pct', 'female_age_10_to_14_pct',
       'female_age_15_to_19_pct', 'female_age_20_to_24_pct',
       'female_age_25_to_29_pct', 'female_age_30_to_34_pct',
       'female_age_35_to_39_pct', 'female_age_40_to_44_pct',
       'female_age_45_to_49_pct', 'female_age_50_to_54_pct',
       'female_age_55_to_59_pct', 'female_age_60_to_64_pct',
       'female_age_65_to_69_pct', 'female_age_70_to_74_pct',
       'female_age_75_to_79_pct', 'female_age_80_to_84_pct',
       'female_age_85_and_over_pct', 'male_total_population',
       'male_age_under_5_pct', 'male_age_5_to_9_pct', 'male_age_10_to_14_pct',
       'male_age_15_to_19_pct', 'male_age_20_to_24_pct',
       'male_age_25_to_29_pct', 'male_age_30_to_34_pct',
       'male_age_35_to_39_pct', 'male_age_40_to_44_pct',
       'male_age_45_to_49_pct', 'male_age_50_to_54_pct',
       'male_age_55_to_59_pct', 'male_age_60_to_64_pct',
       'male_age_65_to_69_pct', 'male_age_70_to_74_pct',
       'male_age_75_to_79_pct', 'male_age_80_to_84_pct',
       'male_age_85_and_over_pct']]
```

In [38]:
```
y = df.county_population_increased_2015_2016
```

In [39]:
```
X
```

Out[39]:

	county name	state	year	female_total_population	female_age_under_5_pct	female_age_5_to_9_pct
0	572	35	2010	192651	10595.805	12137.013
1	580	35	2010	279592	15657.152	16216.336
2	602	35	2010	108490	5858.460	6400.910
3	606	35	2010	47279	2600.345	2458.508
4	627	35	2010	105706	6765.184	7399.420
...	...	...	...	...	...	...
4946	594	39	2015	38559	1773.714	2583.453
4947	595	39	2015	43530	2002.380	3134.160
4948	601	39	2015	36804	1619.376	1803.396
4949	40	39	2015	99486	4377.384	5372.244
4950	371	39	2015	41540	1703.140	1287.740

4945 rows × 41 columns

In [40]:
```
y
```

Out[40]:
```
0       0
1       0
2       0
3       0
4       1
       ..
4946    0
4947    0
4948    0
4949    0
4950    0
Name: county_population_increased_2015_2016, Length: 4945, dtype: int8
```

In [41]:
```
from sklearn.model_selection import train_test_split
```

In [42]:
```
X_train , X_test , y_train, y_test = train_test_split(X, y , test_size = 0.3)
```

In [43]:
```
from sklearn.neighbors import KNeighborsClassifier
```

In [44]:
```
KNN = KNeighborsClassifier()
```

In [45]:
```
KNN.fit(X_train, y_train)
```

Out[45]:
```
KNeighborsClassifier()
```

In [46]:
```python
# KNN.score(X_test, y_test)
```

In [47]:
```python
y_pred = KNN.predict(X_test)
```

In [48]:
```python
kn =accuracy_score(y_test , y_pred)
```

In [49]:
```python
print(kn)
```

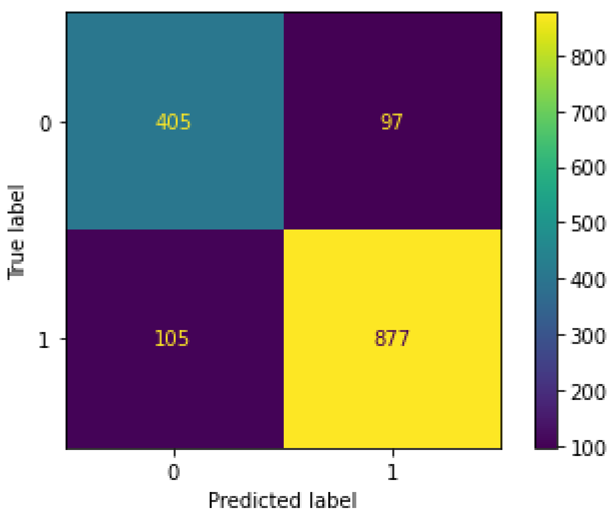
```
0.8638814016172507
```

In [50]:
```python
from sklearn.metrics import f1_score,recall_score,confusion_matrix,precision_scor
```

In [51]:
```python
confusion_matrix(y_test , y_pred)
```

Out[51]:
```
array([[405,  97],
       [105, 877]], dtype=int64)
```

In [52]:
```python
cc = ConfusionMatrixDisplay.from_predictions(y_test , y_pred)
```



In [53]:
```python
f1_score(y_test , y_pred, average='micro')
```

Out[53]:
```
0.8638814016172508
```

In [54]:
```python
f1_score(y_test , y_pred, average='macro')
```

Out[54]:
```
0.8485616366384571
```

In [55]:
```python
recall_score(y_test , y_pred, average='macro')
```

Out[55]:
```
0.8499241323910063
```

In [56]:
```python
precision_score(y_test , y_pred, average='macro')
```

Out[56]:
```
0.8472641623384467
```

In [57]:
```
print(classification_report(y_test , y_pred))
```

```
              precision    recall  f1-score   support

           0       0.79      0.81      0.80       502
           1       0.90      0.89      0.90       982

    accuracy                           0.86      1484
   macro avg       0.85      0.85      0.85      1484
weighted avg       0.86      0.86      0.86      1484
```

In [58]:
```
from sklearn.svm import SVC
```

In [59]:
```
SVM = SVC(kernel ='sigmoid')
```

In [60]:
```
SVM.fit(X_train , y_train)
```

Out[60]:
```
SVC(kernel='sigmoid')
```

In [61]:
```
y_pred = SVM.predict(X_test)
```

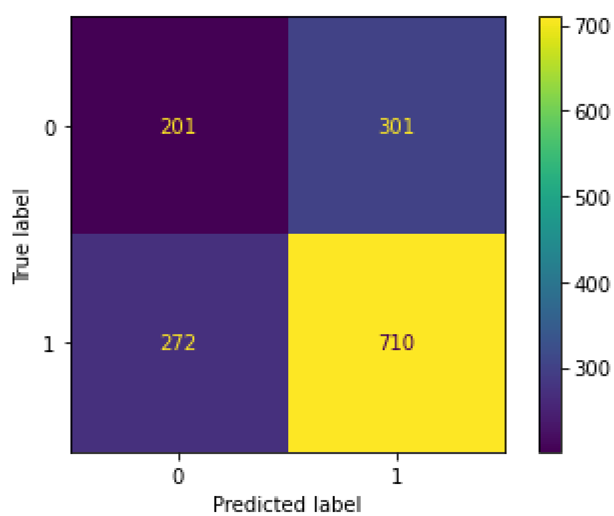
In [62]:
```
sv = accuracy_score(y_test , y_pred)
sv
```

Out[62]:
```
0.6138814016172507
```

In [63]:
```
confusion_matrix(y_test , y_pred)
```

Out[63]:
```
array([[201, 301],
       [272, 710]], dtype=int64)
```

In [64]:
```
cc = ConfusionMatrixDisplay.from_predictions(y_test , y_pred)
```



In [65]:
```
f1_score(y_test , y_pred, average='micro')
```
Out[65]: 0.6138814016172507

In [66]:
```
f1_score(y_test , y_pred, average='macro')
```
Out[66]: 0.5624007101779305

In [67]:
```
recall_score(y_test , y_pred)
```
Out[67]: 0.7230142566191446

In [68]:
```
precision_score(y_test , y_pred)
```
Out[68]: 0.7022749752720079

In [69]:
```
print(classification_report(y_test , y_pred))
```
```
              precision    recall  f1-score   support

           0       0.42      0.40      0.41       502
           1       0.70      0.72      0.71       982

    accuracy                           0.61      1484
   macro avg       0.56      0.56      0.56      1484
weighted avg       0.61      0.61      0.61      1484
```

In [ ]:

In [ ]:

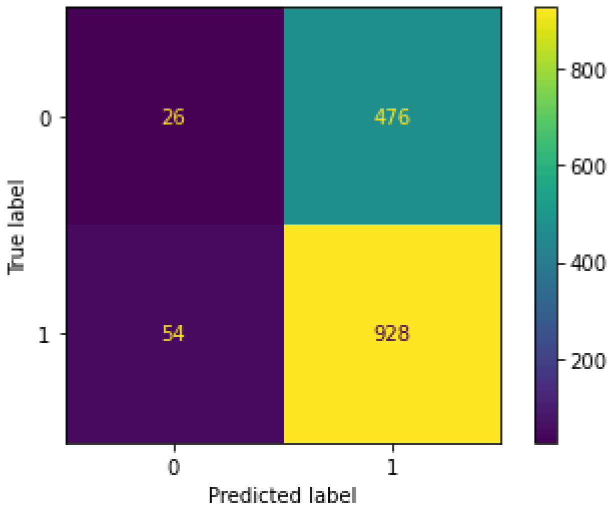
In [70]:
```
from sklearn.naive_bayes import GaussianNB
```
In [71]:
```
GB = GaussianNB()
```
In [72]:
```
GB.fit(X_train , y_train)
```
Out[72]: GaussianNB()

In [73]:
```
y_pred = GB.predict(X_test)
```
In [74]:
```
gb = accuracy_score(y_test ,y_pred )
gb
```
Out[74]: 0.6428571428571429

In [75]:
```
confusion_matrix(y_test , y_pred)
```

Out[75]:
```
array([[ 26, 476],
       [ 54, 928]], dtype=int64)
```

In [76]:
```
cc = ConfusionMatrixDisplay.from_predictions(y_test , y_pred)
```



In [77]:
```
f1_score(y_test , y_pred, average='micro')
```

Out[77]: 0.6428571428571429

In [78]:
```
f1_score(y_test , y_pred, average='macro')
```

Out[78]: 0.4336089963504175

In [79]:
```
recall_score(y_test , y_pred)
```

Out[79]: 0.945010183299389

In [80]:
```
precision_score(y_test , y_pred)
```

Out[80]: 0.6609686609686609

In [81]:
```
print(classification_report(y_test , y_pred))
```

```
              precision    recall  f1-score   support

           0       0.33      0.05      0.09       502
           1       0.66      0.95      0.78       982

    accuracy                           0.64      1484
   macro avg       0.49      0.50      0.43      1484
weighted avg       0.55      0.64      0.54      1484
```

In [ ]:

In [ ]:

```python

```

In [82]:

```python
from sklearn import tree
```

In [83]:

```python
Tree = tree.DecisionTreeClassifier()
```

In [84]:

```python
Tree.fit(X_train , y_train)
```

Out[84]:

```
DecisionTreeClassifier()
```

In [85]:

```python
y_pred = Tree.predict(X_test)
```

In [86]:

```python
tree = accuracy_score(y_test ,y_pred )
tree
```

Out[86]:

```
0.8342318059299192
```

In [87]:

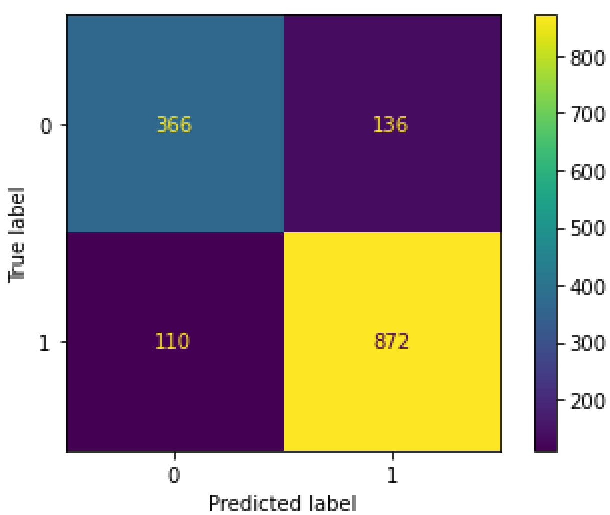
```python
confusion_matrix(y_test , y_pred)
```

Out[87]:

```
array([[366, 136],
       [110, 872]], dtype=int64)
```

In [88]:

```python
cc = ConfusionMatrixDisplay.from_predictions(y_test , y_pred)
```



In [89]:

```python
f1_score(y_test , y_pred, average='micro')
```

Out[89]:

```
0.8342318059299192
```

In [90]:

```python
f1_score(y_test , y_pred, average='macro')
```

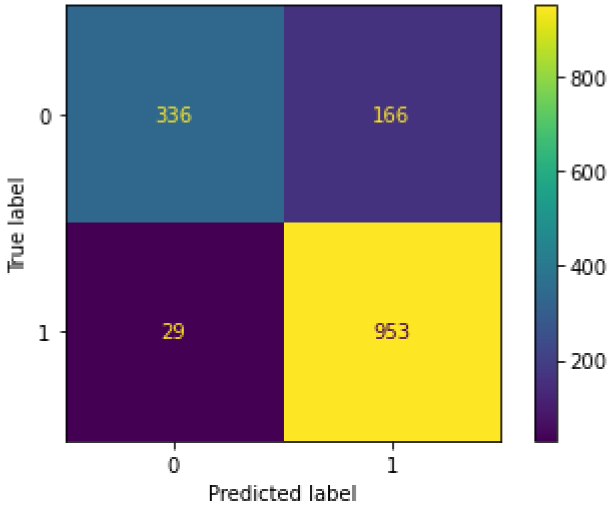
Out[90]:

```
0.8124240836082253
```

In [91]:
```
recall_score(y_test , y_pred)
```
Out[91]:
```
0.8879837067209776
```
In [92]:
```
precision_score(y_test , y_pred)
```
Out[92]:
```
0.8650793650793651
```
In [93]:
```
print(classification_report(y_test , y_pred))
```
```
              precision    recall  f1-score   support

           0       0.77      0.73      0.75       502
           1       0.87      0.89      0.88       982

    accuracy                           0.83      1484
   macro avg       0.82      0.81      0.81      1484
weighted avg       0.83      0.83      0.83      1484
```
In [ ]:

In [94]:
```
from sklearn.ensemble import RandomForestClassifier
```
In [95]:
```
rfc = RandomForestClassifier()
```
In [96]:
```
rfc.fit(X_train , y_train)
```
Out[96]:
```
RandomForestClassifier()
```
In [97]:
```
y_pred = rfc.predict(X_test)
```
In [98]:
```
Rfc = accuracy_score(y_test ,y_pred )
Rfc
```
Out[98]:
```
0.8685983827493261
```
In [99]:
```
confusion_matrix(y_test , y_pred)
```
Out[99]:
```
array([[336, 166],
       [ 29, 953]], dtype=int64)
```

In [100]:
```
cc = ConfusionMatrixDisplay.from_predictions(y_test , y_pred)
```



In [101]:
```
f1_score(y_test , y_pred, average='micro')
```

Out[101]: 0.8685983827493261

In [102]:
```
f1_score(y_test , y_pred, average='macro')
```

Out[102]: 0.8411367794871119

In [103]:
```
recall_score(y_test , y_pred)
```

Out[103]: 0.9704684317718941

In [104]:
```
precision_score(y_test , y_pred)
```

Out[104]: 0.8516532618409294

In [105]:
```
print(classification_report(y_test , y_pred))
```

```
              precision    recall  f1-score   support

           0       0.92      0.67      0.78       502
           1       0.85      0.97      0.91       982

    accuracy                           0.87      1484
   macro avg       0.89      0.82      0.84      1484
weighted avg       0.87      0.87      0.86      1484
```

In [ ]:

In [106]:
```
from sklearn.ensemble import AdaBoostClassifier
```

In [107]:
```
Ada = AdaBoostClassifier()
```

In [108]:
```
Ada.fit(X_train , y_train)
```

Out[108]:
```
AdaBoostClassifier()
```

In [109]:
```
y_pred = Ada.predict(X_test)
```

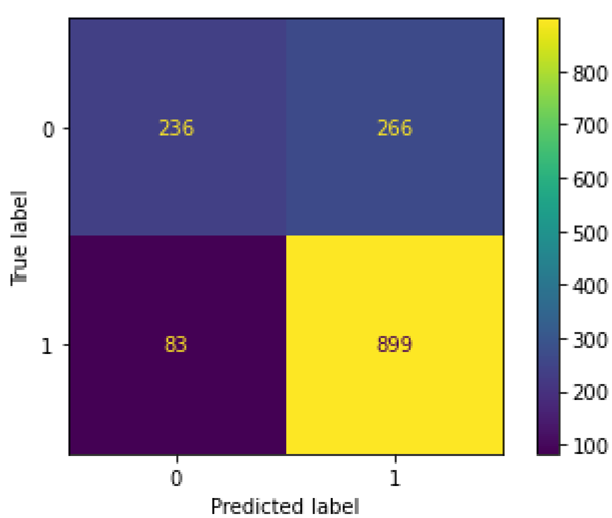
In [110]:
```
Ada_acc = accuracy_score(y_test ,y_pred )
Ada_acc
```

Out[110]:
```
0.7648247978436657
```

In [111]:
```
confusion_matrix(y_test , y_pred)
```

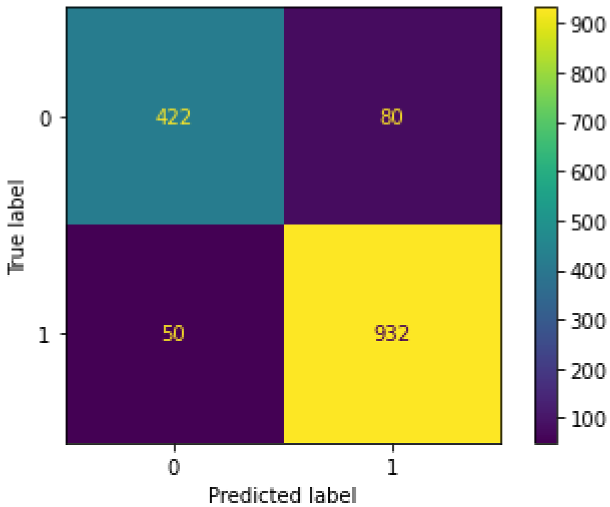
Out[111]:
```
array([[236, 266],
       [ 83, 899]], dtype=int64)
```

In [112]:
```
cc = ConfusionMatrixDisplay.from_predictions(y_test , y_pred)
```



In [113]:
```
f1_score(y_test , y_pred, average='micro')
```
Out[113]:
```
0.7648247978436657
```
In [114]:
```
f1_score(y_test , y_pred, average='macro')
```
Out[114]:
```
0.7061781246472006
```
In [115]:
```
recall_score(y_test , y_pred)
```
Out[115]:
```
0.9154786150712831
```
In [116]:
```
precision_score(y_test , y_pred)
```
Out[116]:
```
0.7716738197424893
```
In [117]:
```
print(classification_report(y_test , y_pred))
```
```
              precision    recall  f1-score   support

           0       0.74      0.47      0.57       502
           1       0.77      0.92      0.84       982

    accuracy                           0.76      1484
   macro avg       0.76      0.69      0.71      1484
weighted avg       0.76      0.76      0.75      1484
```
In [ ]:
```

```
In [118]:
```
from sklearn.ensemble import BaggingClassifier
```
In [119]:
```
ba = BaggingClassifier()
```
In [120]:
```
ba.fit(X_train , y_train)
```
Out[120]:
```
BaggingClassifier()
```
In [121]:
```
y_pred = ba.predict(X_test)
```
In [122]:
```
Ba_acc = accuracy_score(y_test ,y_pred )
Ba_acc
```
Out[122]:
```
0.9123989218328841
```
In [123]:
```
confusion_matrix(y_test , y_pred)
```
Out[123]:
```
array([[422,  80],
       [ 50, 932]], dtype=int64)
```

In [124]:
```python
cc = ConfusionMatrixDisplay.from_predictions(y_test , y_pred)
```



In [125]:
```python
f1_score(y_test , y_pred, average='micro')
```

Out[125]: 0.9123989218328841

In [126]:
```python
f1_score(y_test , y_pred, average='macro')
```

Out[126]: 0.9006670936835146

In [127]:
```python
recall_score(y_test , y_pred)
```

Out[127]: 0.9490835030549898

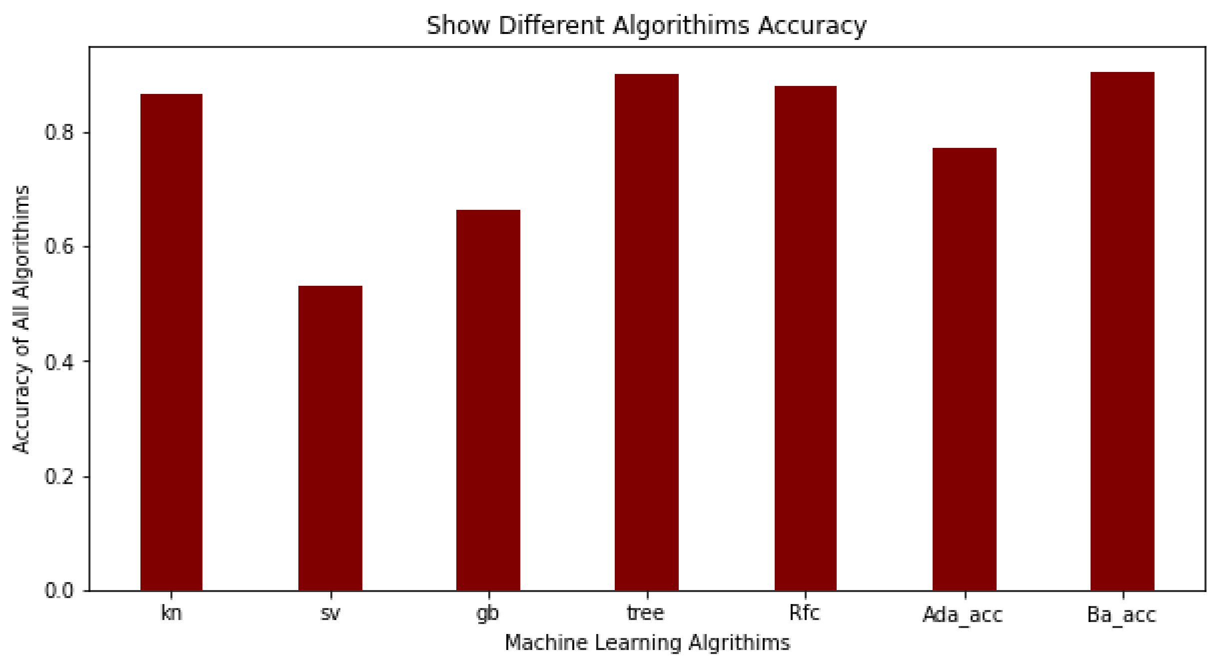
In [128]:
```python
precision_score(y_test , y_pred)
```

Out[128]: 0.9209486166007905

In [129]:
```python
print(classification_report(y_test , y_pred))
```

```
              precision    recall  f1-score   support

           0       0.89      0.84      0.87       502
           1       0.92      0.95      0.93       982

    accuracy                           0.91      1484
   macro avg       0.91      0.89      0.90      1484
weighted avg       0.91      0.91      0.91      1484
```

In [130]:
```
data = {'kn':0.8672506738544474, 'sv':0.532345013477089, 'gb':0.6650943396226415,
        'tree':0.9022911051212938, 'Rfc':0.8793800539083558,'Ada_acc':0.773584905

fig = plt.figure(figsize = (10, 5))

# creating the bar plot
courses = list(data.keys())
values = list(data.values())
plt.bar(courses, values, color ='maroon',
        width = 0.4)

plt.xlabel("Machine Learning Algrithims")
plt.ylabel("Accuracy of All Algorithims")
plt.title("Show Different Algorithims Accuracy")
plt.show()
```



In [131]:
```
# Hyperparameter tunning
```

In [132]:
```
# from sklearn.model_selection import GridSearchCV
# ba = BaggingClassifier(ba, n_estimators = 500, max_samples = 0.8, max_features
# ba = ba.fit(X_train, y_train)
```

In [133]:
```
# y_pred = ba.predict(X_test)
```

In [134]:
```python
# Ba_acc = accuracy_score(y_test ,y_pred )
# Ba_acc
```

Out[134]:
```
0.8483827493261455
```

In [141]:
```python
from sklearn.model_selection import GridSearchCV

# defining parameter range
param_grid = {'C': [0.1, 1, 10, 100, 1000],
              'gamma': [1, 1.0, 0.01, 0.001, 0.0001],
              'kernel': ['rbf']}

grid = GridSearchCV(SVC(), param_grid, refit = True, verbose = 3)

# fitting the model for grid search
grid.fit(X_train, y_train)
```

```
Fitting 5 folds for each of 25 candidates, totalling 125 fits
[CV 1/5] END ........C=0.1, gamma=1, kernel=rbf;, score=0.677 total time=
5.4s
[CV 2/5] END ........C=0.1, gamma=1, kernel=rbf;, score=0.676 total time=
4.4s
[CV 3/5] END ........C=0.1, gamma=1, kernel=rbf;, score=0.676 total time=
4.3s
[CV 4/5] END ........C=0.1, gamma=1, kernel=rbf;, score=0.676 total time=
4.3s
[CV 5/5] END ........C=0.1, gamma=1, kernel=rbf;, score=0.676 total time=
4.3s
[CV 1/5] END ......C=0.1, gamma=1.0, kernel=rbf;, score=0.677 total time=
4.3s
[CV 2/5] END ......C=0.1, gamma=1.0, kernel=rbf;, score=0.676 total time=
4.3s
[CV 3/5] END ......C=0.1, gamma=1.0, kernel=rbf;, score=0.676 total time=
4.3s
[CV 4/5] END ......C=0.1, gamma=1.0, kernel=rbf;, score=0.676 total time=
4.4s
```

In [142]:
```python


# print best parameter after tuning
print(grid.best_params_)

# print how our model looks after hyper-parameter tuning
print(grid.best_estimator_)
```

```
{'C': 0.1, 'gamma': 1, 'kernel': 'rbf'}
SVC(C=0.1, gamma=1)
```

In [143]:
```
y_pred = SVM.predict(X_test)

# print classification report
print(classification_report(y_test, y_pred ))
```

```
              precision    recall  f1-score   support

           0       0.42      0.40      0.41       502
           1       0.70      0.72      0.71       982

    accuracy                           0.61      1484
   macro avg       0.56      0.56      0.56      1484
weighted avg       0.61      0.61      0.61      1484
```

In [144]:
```
Svm_acc = accuracy_score(y_test ,y_pred )
Svm_acc
```

Out[144]:
```
0.6138814016172507
```

In [ ]:
```

```