**Banks in Distress: Analyzing the Regions of US Bank Failures**

**Data Cleaning Process:**

***Step 1: #* Import all necessary API’s**

**import** bankfind **as** bf

**import** pandas **as** pd

**import** folium

**from** folium.plugins **import** HeatMap

**import** geopandas **as** gpd

**from** pprint **import** pprint

**import** matplotlib.pyplot **as** plt

**import** requests

**Step 2: # Retrieve data on failed financial institutions.**

**FDIC allows direct import of different modules. This one just asks for failed banks.**

data **=** bf**.**get\_failures()

*# Inspect the structure of the data*

print(type(data))

**Step 3:** # **This is what a single data entry for a failed bank looks like.**

**Note: We'll need to clean this up, reorder it.**

data['data'][0]

{'QBFDEP': 64941,

'PSTALP': 'KS',

'FIN': '10538',

'FAILDATE': '10/23/2020',

'RESTYPE': 'FAILURE',

'CITYST': 'ALMENA, KS',

'SAVR': 'DIF',

'RESTYPE1': 'PA',

'CHCLASS1': 'NM',

'NAME': 'ALMENA STATE BANK',

'COST': 16806,

'QBFASSET': 65733,

'CERT': 15426,

'FAILYR': '2020',

'ID': '4104'}

In [84]:

**Step 4: # Create a data frame that hosts all data for each failed bank.**

**This will need to be cleaned up.**

data **=** bf**.**get\_failures()

df **=** pd**.**DataFrame(data['data'])

df

Out[84]:

|  | **QBFDEP** | **PSTALP** | **FIN** | **FAILDATE** | **RESTYPE** | **CITYST** | **SAVR** | **RESTYPE1** | **CHCLASS1** | **NAME** | **COST** | **QBFASSET** | **CERT** | **FAILYR** | **ID** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | 64941.0 | KS | 10538 | 10/23/2020 | FAILURE | ALMENA, KS | DIF | PA | NM | ALMENA STATE BANK | 16806.0 | 65733.0 | 15426.0 | 2020 | 4104 |
| **1** | 133936.0 | FL | 10537 | 10/16/2020 | FAILURE | FORT WALTON BEACH, FL | DIF | PA | NM | FIRST CITY BANK OF FLORIDA | 7247.0 | 136566.0 | 16748.0 | 2020 | 4103 |
| **2** | 143102.0 | WV | 10536 | 4/3/2020 | FAILURE | BARBOURSVILLE, WV | DIF | PA | NM | THE FIRST STATE BANK | 45913.0 | 151808.0 | 14361.0 | 2020 | 4102 |
| **3** | 95159.0 | NE | 10535 | 2/14/2020 | FAILURE | ERICSON, NE | DIF | PA | NM | ERICSON STATE BANK | 25293.0 | 100879.0 | 18265.0 | 2020 | 4101 |
| **4** | 111234.0 | NJ | 10534 | 11/1/2019 | FAILURE | NEWARK, NJ | DIF | PA | N | CITY NATIONAL BANK OF NEW JERSEY | 1946.0 | 120574.0 | 21111.0 | 2019 | 4100 |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| **4099** | 68.0 | KY | 0 | 8/6/1934 | FAILURE | LEWISPORT, KY | BIF | PO | NM | BANK OF LEWISPORT | NaN | 81.0 | NaN | 1934 | 137 |
| **4100** | 42.0 | MT | 0 | 7/18/1934 | FAILURE | LIMA, MT | BIF | PO | N | FIRST NATIONAL BANK OF LIMA | NaN | 91.0 | NaN | 1934 | 135 |
| **4101** | 69.0 | IN | 0 | 7/18/1934 | FAILURE | FLORENCE, IN | BIF | PO | NM | FLORENCE DEPOSIT BANK | NaN | 105.0 | NaN | 1934 | 136 |
| **4102** | 238.0 | IL | 0 | 5/28/1934 | FAILURE | EAST PEORIA, IL | BIF | PO | NM | FON DU LAC STATE BANK | NaN | 374.0 | NaN | 1934 | 133 |
| **4103** | 1064.0 | PA | 0 | 4/19/1934 | FAILURE | PITTSBURGH, PA | BIF | PO | NM | BANK OF AMERICA TRUST CO. | NaN | 1435.0 | NaN | 1934 | 134 |

**Step 5:**  **Rename the columns to the following categories:**

**Deposits, State, Fed RSSD, Failure Date, Resolution Type, City/State, Savr, Resolution Type 1, Charter Class 1, Bank Name , Cost, Assets, FDIC, Failure Year, ID**

new\_columns = {

'QBFDEP': 'Deposits',

'PSTALP': 'State',

'FIN': 'Fed RSSD',

'FAILDATE': 'Failure Date',

'RESTYPE': 'Resolution Type',

'CITYST': 'City/State',

'SAVR': 'SAVR',

'RESTYPE1': 'Resolution Type 1',

'CHCLASS1': 'Charter Class 1',

'NAME': 'Bank Name',

'COST': 'Cost',

'QBFASSET': 'Assets',

'CERT': 'FDIC Cert',

'FAILYR': 'Failure Year',

'ID': 'ID'

}

df = df.rename(columns=new\_columns)

**Step 5: Reorder the columns, creates a loop that arbitrarily lists the remaining columns after a specified order**

df = df[['Bank Name', 'ID', 'State', 'Failure Date', 'Failure Year', 'Assets', 'Deposits']+ [c for c in df.columns if c not in ['Bank Name', 'ID', 'State', 'Failure Date', 'Failure Year', 'Assets', 'Deposits']]]

df.head()

|  | **Bank Name** | **ID** | **State** | **Failure Date** | **Failure Year** | **Assets** | **Deposits** | **Fed RSSD** | **Resolution Type** | **City/State** | **SAVR** | **Resolution Type 1** | **Charter Class 1** | **Cost** | **FDIC Cert** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | ALMENA STATE BANK | 4104 | KS | 10/23/2020 | 2020 | 65733.0 | 64941.0 | 10538 | FAILURE | ALMENA, KS | DIF | PA | NM | 16806.0 | 15426.0 |
| **1** | FIRST CITY BANK OF FLORIDA | 4103 | FL | 10/16/2020 | 2020 | 136566.0 | 133936.0 | 10537 | FAILURE | FORT WALTON BEACH, FL | DIF | PA | NM | 7247.0 | 16748.0 |
| **2** | THE FIRST STATE BANK | 4102 | WV | 4/3/2020 | 2020 | 151808.0 | 143102.0 | 10536 | FAILURE | BARBOURSVILLE, WV | DIF | PA | NM | 45913.0 | 14361.0 |
| **3** | ERICSON STATE BANK | 4101 | NE | 2/14/2020 | 2020 | 100879.0 | 95159.0 | 10535 | FAILURE | ERICSON, NE | DIF | PA | NM | 25293.0 | 18265.0 |
| **4** | CITY NATIONAL BANK OF NEW JERSEY | 4100 | NJ | 11/1/2019 | 2019 | 120574.0 | 111234.0 | 10534 | FAILURE | NEWARK, NJ | DIF | PA | N | 1946.0 | 21111.0 |

**Step 6: # import matplotlib.ticker as mtick**

**Step 7: # Group the data by year and count the number of banks that failed each year**

failures\_per\_year = df.groupby('Failure Year').size()

**Step 8: Group the data by year and calculate the total cost of failures each year**

cost\_per\_year = df.groupby('Failure Year')['Cost'].sum()

**Step: 9 # Create a bar chart of the data**

fig, ax = plt.subplots()

ax.bar(failures\_per\_year.index, failures\_per\_year.values)

**Step 10: # Create a second y-axis for the cost data**

ax2 = ax.twinx()

**Step 11: # Create a line chart of the cost data, with the color set to green**

**ax2.plot(cost\_per\_year.index, cost\_per\_year.values, color='green')**

**Step 12: # Set the chart title and axis labels**

ax.set\_title('Number of Failed Banks and Cost by Year')

ax.set\_xlabel('Year')

ax.set\_ylabel('Number of Failed Banks')

**Step 13: # Set the y-axis label for the cost data to be in dollars**

fmt = '${x:,.0f}'

tick = mtick.StrMethodFormatter(fmt)

ax2.yaxis.set\_major\_formatter(tick)

**Step 14: # Set the x-axis tick locations and labels to show only every 10 years, starting at the nearest multiple of 10 greater than or equal to 1950**

start\_year = (1950 // 10) \* 10

ax.set\_xticks(range(start\_year, failures\_per\_year.index[-1]+1, 10))

**Step 15: # Add grid lines to the chart**

ax.grid(True, axis='y')

ax2.grid(False)

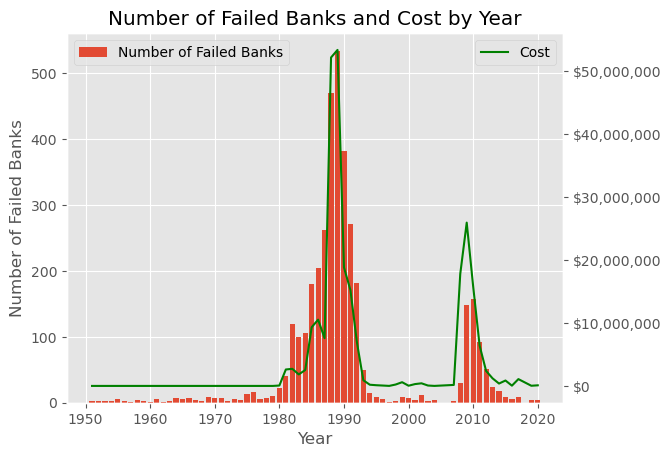
**Step 16: # Add a legend for the two y-axes**

ax.legend(['Number of Failed Banks'], loc='upper left')

ax2.legend(['Cost'], loc='upper right')

**Step 17: # Display the chart**

**plt.show()**



**In [186]:**

**Step 18: Group the data by year and count the number of banks that failed each year**

failures\_per\_year **=** df**.**groupby('Failure Year')**.**size()

***Step 19: # Create a line chart of the data***

plt**.**plot(failures\_per\_year**.**index, failures\_per\_year**.**values, color**=**'red')

**Step 20: # Create a bar chart of the data**

plt**.**bar(failures\_per\_year**.**index, failures\_per\_year**.**values, color**=**'blue')

**Step 21: # Set the chart title and axis labels**

plt**.**title('Number of Failed Banks by Year')

plt**.**xlabel('Year')

plt**.**ylabel('Number of Failed Banks')

**Step 22: # Set the x-axis tick locations and labels to show only every 5 years between 1975 and 1995**

plt**.**xticks(range(1975, 1996, 5))

**Step 23: # Limit the x-axis range to only show between 1975 and 1995**

plt**.**xlim(1975, 1995)

**Step 24: # Display the chart**

plt**.**show()

Chart, histogram

Description automatically generated

## Bank Failures in Brief – Summary 80s through 90s

"The rise in the number of bank failures in the 1980s had no single cause or short list of causes. Rather, it resulted from a concurrence of various forces working to gether to produce a decade of banking crises." <https://www.fdic.gov/analysis/archived-research/banking-review/brspecial.pdf>

**In [185]:**

**Step 25: # Filter the data to only include years between 2000 and 2023**

failures\_per\_year **=** failures\_per\_year**.**loc[2000:2023]

**Step 26: # Create a bar chart of the data and set the color to blue**

plt**.**bar(failures\_per\_year**.**index, failures\_per\_year**.**values, color**=**'blue')

**Step 27: # Create a line chart of the data and set the color to red**

plt**.**plot(failures\_per\_year**.**index, failures\_per\_year**.**values, color**=**'red')

**Step 28: # Set the chart title and axis labels**

plt**.**title('Number of Failed Banks by Year')

plt**.**xlabel('Year')

plt**.**ylabel('Number of Failed Banks')

**Step 29: # Set the x-axis tick locations and labels to show only every 5 years between 2000 and 2023**

plt**.**xticks(range(2000, 2024, 5), range(2000, 2024, 5))

**Step 30: # Display the chart**

plt**.**show()

Chart, histogram

Description automatically generated

## Bank Failures in Brief – Summary 2001 through 2023

<https://www.fdic.gov/bank/historical/bank/>

In [90]:

**Step 31: # Get the top 10 states by number of failed banks**

top\_10\_states **=** df**.**groupby('State')**.**size()**.**sort\_values(ascending**=False**)**.**head(10)

**Step 32: # Create a pie chart of the top 10 states**

plt**.**pie(top\_10\_states, labels**=**top\_10\_states**.**index, autopct**=**'%1.1f%%')

**Step 33: # Set the chart title**

plt**.**title('Top 10 States by Number of Failed Banks')

**Step 34: # Display the chart**

plt**.**show()

Chart, pie chart

Description automatically generated

In [92]:

**Step 35: # Filter the data for years 1975-1995**

df\_75\_95 **=** df[(df['Failure Year'] **>=** 1975) **&** (df['Failure Year'] **<=** 1995)]

**Step 36: # Group by state and count the number of failed banks**

top\_10\_states\_75\_95 **=** df\_75\_95**.**groupby('State')**.**size()**.**sort\_values(ascending**=False**)**.**head(10)

**Step 37: # Create a horizontal bar chart for the top 10 states for years 1975-1995**

fig, ax **=** plt**.**subplots(figsize**=**(8, 6))

ax**.**barh(top\_10\_states\_75\_95**.**index, top\_10\_states\_75\_95**.**values, color**=**'red')

ax**.**set\_title('Top 10 States by Number of Failed Banks (1975-1995)')

ax**.**set\_xlabel('Number of Failed Banks')

ax**.**set\_ylabel('State')

**Step 38: # Filter the data for years 2000-2020**

df\_00\_20 **=** df[(df['Failure Year'] **>=** 2000) **&** (df['Failure Year'] **<=** 2020)]

**Step 39: # Group by state and count the number of failed banks**

top\_10\_states\_00\_20 **=** df\_00\_20**.**groupby('State')**.**size()**.**sort\_values(ascending**=False**)**.**head(10)

**Step 40: # Create a horizontal bar chart for the top 10 states for years 2000-2020**

fig, ax **=** plt**.**subplots(figsize**=**(8, 6))

ax**.**barh(top\_10\_states\_00\_20**.**index, top\_10\_states\_00\_20**.**values, color**=**'blue')

ax**.**set\_title('Top 10 States by Number of Failed Banks (2000-2020)')

ax**.**set\_xlabel('Number of Failed Banks')

ax**.**set\_ylabel('State')

**Step 41: # Display the charts**

plt**.**show()

Chart, bar chart

Description automatically generated

Chart, bar chart

Description automatically generated

In [156]:

**Step 42: import** plotly.express **as** px

*Step 43: # Count the number of failed banks per state*

state\_count **=** df**.**groupby('State')['Bank Name']**.**count()**.**reset\_index(name**=**'Failures')

**Step 43: # Create a choropleth map figure**

fig **=** px**.**choropleth(

state\_count,

locations**=**'State', *# state codes*

color**=**'Failures', *# number of failures*

locationmode**=**'USA-states', *# set location mode to USA states*

scope**=**'usa', *# set scope to USA*

color\_continuous\_scale**=**'Blues', *# set colorscale*

title**=**'Number of Failed Banks by State (1950-2020)'

)

**Step 44: # Set figure size**

fig**.**update\_layout(width**=**800, height**=**500)

*# Display the figure*

fig**.**show()

In [94]:

new\_df **=** df[['Bank Name', 'City/State', 'Failure Year']]

**Step 45:** # set Geoapify API key

API\_KEY = "5ea46e0aae3948818dce6e65b8156ab3"

BASE\_URL = "https://api.geoapify.com/v1/geocode/search"

**Step 46:** # create empty lists to store the latitude and longitude values

latitudes = []

longitudes = []

**Step 47:** # loop through each row in the dataframe and geocode the city/state

for index, row in new\_df.iterrows():

location = f"{row['City/State']}, USA"

params = {"text": location, "apiKey": API\_KEY}

response = requests.get(BASE\_URL, params=params).json()

**Step 48: # check if any results were found**

if response["features"]:

# get the first result and extract the latitude and longitude

coordinates = response["features"][0]["geometry"]["coordinates"]

latitudes.append(coordinates[1])

longitudes.append(coordinates[0])

else:

# if no results were found, add NaN values to the lists

latitudes.append(float("NaN"))

longitudes.append(float("NaN"))

**Step 49:**  **# print a message every 50 records**

if index % 50 == 0:

print(f"Processed {index} records...")

**Step 50: # create a new dataframe**

new\_df\_100 = new\_df.copy()

**Step 51:** # add the latitude and longitude columns to the new dataframe

new\_df\_100['Latitude'] = latitudes

new\_df\_100['Longitude'] = longitudes

new\_df\_100.head(5)

Processed 0 records...

Processed 50 records...

Processed 100 records...

Processed 150 records...

Processed 200 records...

Processed 250 records...

Processed 300 records...

Processed 350 records...

Processed 400 records...

Processed 450 records...

Processed 500 records...

Processed 550 records...

Processed 600 records...

Processed 650 records...

Processed 700 records...

Processed 750 records...

Processed 800 records...

Processed 850 records...

Processed 900 records...

Processed 950 records...

Processed 1000 records...

Processed 1050 records...

Processed 1100 records...

Processed 1150 records...

Processed 1200 records...

Processed 1250 records...

Processed 1300 records...

Processed 1350 records...

Processed 1400 records...

Processed 1450 records...

Processed 1500 records...

Processed 1550 records...

Processed 1600 records...

Processed 1650 records...

Processed 1700 records...

Processed 1750 records...

Processed 1800 records...

Processed 1850 records...

Processed 1900 records...

Processed 1950 records...

Processed 2000 records...

Processed 2050 records...

Processed 2100 records...

Processed 2150 records...

Processed 2200 records...

Processed 2250 records...

Processed 2300 records...

Processed 2350 records...

Processed 2400 records...

Processed 2450 records...

Processed 2500 records...

Processed 2550 records...

Processed 2600 records...

Processed 2650 records...

Processed 2700 records...

Processed 2750 records...

Processed 2800 records...

Processed 2850 records...

Processed 2900 records...

Processed 2950 records...

Processed 3000 records...

Processed 3050 records...

Processed 3100 records...

Processed 3150 records...

Processed 3200 records...

Processed 3250 records...

Processed 3300 records...

Processed 3350 records...

Processed 3400 records...

Processed 3450 records...

Processed 3500 records...

Processed 3550 records...

Processed 3600 records...

Processed 3650 records...

Out[95]:

|  | **Bank Name** | **City/State** | **Failure Year** | **Latitude** | **Longitude** |
| --- | --- | --- | --- | --- | --- |
| **0** | ALMENA STATE BANK | ALMENA, KS | 2020 | 39.892229 | -99.707058 |
| **1** | FIRST CITY BANK OF FLORIDA | FORT WALTON BEACH, FL | 2020 | 30.405755 | -86.618842 |
| **2** | THE FIRST STATE BANK | BARBOURSVILLE, WV | 2020 | 38.409530 | -82.294592 |
| **3** | ERICSON STATE BANK | ERICSON, NE | 2020 | 41.779924 | -98.678582 |
| **4** | CITY NATIONAL BANK OF NEW JERSEY | NEWARK, NJ | 2019 | 40.735657 | -74.172367 |

**Step 52:** *# create a map centered on the US*

map\_usa **=** folium.Map(location**=**[39.8283, **-**98.5795], zoom\_start**=**4)

​

**Step 53:** *# add a marker for each location*

**for** index, row **in** new\_df\_100.dropna(subset**=**['Latitude', 'Longitude']).iterrows():

folium.CircleMarker(

location**=**[row['Latitude'], row['Longitude']],

radius**=**1,

color**=**'blue',

fill\_color**=**'blue',

fill\_opacity**=**0.9, *# change the fill opacity*

popup**=**row['City/State'],

).add\_to(map\_usa)

​

**Step 54:** *# display the map*

map\_usa

Out[97]:

In



**Step 55: # create a map centered on the US**

map\_usa **=** folium.Map(location**=**[39.8283, **-**98.5795], zoom\_start**=**4)

​

**Step 56:** *# add a marker for each location*

**for** index, row **in** new\_df\_100.dropna(subset**=**['Latitude', 'Longitude']).iterrows():

folium.CircleMarker(

location**=**[row['Latitude'], row['Longitude']],

radius**=**1,

color**=**'blue',

fill\_color**=**'blue',

fill\_opacity**=**0.9, *# change the fill opacity*

popup**=**row['City/State'],

).add\_to(map\_usa)

​

**Step 57:** *# display the map*

map\_usa

Out[97]:

**Step 58:** *# create a map centered on the US*

map\_usa **=** folium.Map(location**=**[39.8283, **-**98.5795], zoom\_start**=**4)

​

**Step 59: # create LayerControl object to allow for toggling between layers**

layer\_control **=** folium.LayerControl(position**=**'topright')

​

**Step 60 # add a marker for each location**

**for** index, row **in** new\_df\_100.iterrows():

**Step 61:**  **# check if the year is between 2000-2020**

**if** 2000 **<=** row['Failure Year'] **<=** 2020:

*# if yes, make the color orange*

color **=** 'orange'

*# check if the bank closing year is between 1975-1995*

**elseif** 1975 **<=** row['Failure Year'] **<=** 1995:

*# if yes, make the color blue*

color **=** 'blue'

**else**:

*# otherwise, make it gray*

color **=** 'gray'

**Step 62: # check if the Latitude or Longitude values are NaN**

**if** **not** pd.isna(row['Latitude']) **and** **not** pd.isna(row['Longitude']):

folium.CircleMarker(

location**=**[row['Latitude'], row['Longitude']],

radius**=**1,

color**=**color, *# use the color variable based on the Failure Year and Closing Year*

fill\_color**=**color, *# use the same color for the fill*

fill\_opacity**=**0.9,

popup**=**row['City/State'],

).add\_to(map\_usa)

​

**Step 63: # create custom HTML content for the legend**

legend\_html **=** '''

<div style="position: fixed;

bottom: 50px; left: 50px; width: 200px; height: 100px;

border:2px solid grey; z-index:9999; font-size:14px;

background-color:rgba(255, 255, 255, 0.7);

">

<p style="margin: 5px; color: blue;">1975-1995</p>

<p style="margin: 5px; color: orange;">2000-2020</p>

<p style="margin: 5px; color: grey;">Other</p>

</div>

'''

​

**Step 64:** *# create a Marker object with the legend HTML content and add it to the map*

legend\_marker **=** folium.Marker(location**=** [45.2538, **-**69.4455], icon**=**folium.Icon(color**=**'white', icon**=**'info-sign'))

legend\_marker.add\_child(folium.Popup(legend\_html))

map\_usa.add\_child(legend\_marker)

​

**Step 65:** *# add the LayerControl to the map*

map\_usa.add\_child(layer\_control)

​

**Step 66:** *# display the map*

map\_usa

Out[162]:

Chart, map, bubble chart

Description automatically generated

In [241]:

​

**Step 67: # create a list of points as [latitude, longitude] pairs for all years**

locations\_all **=** new\_df\_100[['Latitude', 'Longitude']]

​

**Step 68: # cluster the data using K-Means**

kmeans **=** KMeans(n\_clusters**=**10, random\_state**=**42).fit(locations\_all)

​

**Step 69: # add cluster labels to the data**

locations\_all **=** locations\_all.assign(cluster**=**kmeans.labels\_)

​

**Step 70: # create a map**

map\_clusters **=** folium.Map(location**=**[39.8283, **-**98.5795], zoom\_start**=**4)

​

**Step 71:** *# create a MarkerCluster for each cluster*

**for** i **in** range(kmeans.n\_clusters):

*# filter the data for this cluster*

cluster\_data **=** locations\_all[locations\_all['cluster'] **==** i]

*# count the number of banks in this cluster*

num\_banks **=** len(cluster\_data)

*# create a MarkerCluster for this cluster*

cluster\_marker **=** MarkerCluster(name**=**f"Cluster {i**+**1} ({num\_banks} banks)").add\_to(map\_clusters)

*# add markers for each bank in this cluster*

**for** \_, row **in** cluster\_data.iterrows():

folium.Marker(location**=**[row['Latitude'], row['Longitude']]).add\_to(cluster\_marker)

​

**Step 72: # add layer control to the map**

folium.LayerControl().add\_to(map\_clusters)

​

**Step 73: # display the map**

display(map\_clusters)

all\_years\_df **=** new\_df\_100.copy()

​

**Step 74: # create dataframe for 1980-1995**

df\_1980\_1995 **=** new\_df\_100[(new\_df\_100['Failure Year'] **>=** 1980) **&** (new\_df\_100['Failure Year'] **<=** 1995)].copy()

​

**Step 75: # create dataframe for 2000-2020**

df\_2000\_2020 **=** new\_df\_100[new\_df\_100['Failure Year'].between(2000, 2020)].copy()

map\_all\_years **=** folium.Map(location**=**[39.8283, **-**98.5795], zoom\_start**=**4)

map\_1980\_1995 **=** folium.Map(location**=**[30.2855, **-**97.6654], zoom\_start**=**4)

map\_2000\_2020 **=** folium.Map(location**=**[39.8283, **-**98.5795], zoom\_start**=**4)

​

**Step 76:** *# create a list of points as [latitude, longitude] pairs for all years*

locations\_all **=** new\_df\_100[['Latitude', 'Longitude']].values.tolist()

​

**Step 77:***# create the heat map*

heat\_layer\_all **=** HeatMap(locations\_all)

map\_all\_years.add\_child(heat\_layer\_all)

​

**Step 78:***# create a list of points as [latitude, longitude] pairs for 1980-1995*

df\_1980\_1995 = new\_df\_100[new\_df\_100['Failure Year'].between(1980, 1995)]

locations\_1980\_1995 = df\_1980\_1995[['Latitude', 'Longitude']].values.tolist()

​

​

heat\_layer\_1980\_1995 = HeatMap(locations\_1980\_1995)

map\_1980\_1995.add\_child(heat\_layer\_1980\_1995)

​

**Step 79:*# create a list of points [latitude, longitude] for 2000-2020***

df\_2000\_2020 **=** new\_df\_100[new\_df\_100['Failure Year'].between(2000, 2020)]

locations\_2000\_2020 **=** df\_2000\_2020[['Latitude', 'Longitude']].values.tolist()

​

​

heat\_layer\_2000\_2020 **=** HeatMap(locations\_2000\_2020)

map\_2000\_2020.add\_child(heat\_layer\_2000\_2020)

​

display(map\_all\_years)

display(map\_1980\_1995)

display(map\_2000\_2020)

Map

Description automatically generated

**Step 80: # select only the columns you want to keep**

new\_df **=** df[['Bank Name', 'State', 'Failure Year', 'Assets', 'Deposits']].copy()

​

**Step 81: *# create a new column 'Difference' that calculates the difference between 'Deposits' and 'Assets' only where 'Deposits' are larger than 'Assets'***

new\_df.loc[new\_df['Deposits'] **>** new\_df['Assets'], 'Difference'] **=** new\_df['Deposits'] **-** new\_df['Assets']

new\_df **=** new\_df.sort\_values(by**=**'Difference', ascending**=False**)

​

**Step 82: # drop rows with NaN values in 'Difference' column**

new\_df **=** new\_df.dropna(subset**=**['Difference'])

​

**Step 83: *# display the new dataframe***

**new\_df**

Out[134]:

|  | **Bank Name** | **State** | **Failure Year** | **Assets** | **Deposits** | **Difference** |
| --- | --- | --- | --- | --- | --- | --- |
| **2230** | SUNBELT SAVINGS ASSOCIATION OF TEXAS | TX | 1988 | 2214129.0 | 3255324.0 | 1041195.0 |
| **2075** | COMMODORE SAVINGS ASSOCIATION | TX | 1988 | 499435.0 | 1139370.0 | 639935.0 |
| **1988** | SECURITY S & LA | AZ | 1989 | 626922.0 | 1052619.0 | 425697.0 |
| **1778** | MERIDIAN SAVING ASSOCIATION | TX | 1989 | 48872.0 | 434862.0 | 385990.0 |
| **2666** | INDEPENDENT AMERICAN SA | TX | 1987 | 1014023.0 | 1392604.0 | 378581.0 |
| **...** | ... | ... | ... | ... | ... | ... |
| **671** | COLUMBIA NATIONAL BANK | CA | 1993 | 47618.0 | 47627.0 | 9.0 |
| **812** | SOUTHSIDE NATIONAL BANK | TX | 1992 | 12040.0 | 12049.0 | 9.0 |
| **481** | CITIZENS COMMUNITY BANK | NJ | 2009 | 40657.0 | 40664.0 | 7.0 |
| **3688** | BRAZEAU BANK | MO | 1951 | 144.0 | 148.0 | 4.0 |
| **2684** | THE PEOPLES BANK | AL | 1987 | 11896.0 | 11899.0 | 3.0 |

905 rows × 6 columns

state\_difference **=** new\_df.groupby('State')['Difference'].sum().reset\_index()

state\_difference['Difference'] **=** state\_difference['Difference'].abs()

state\_difference **=** state\_difference.sort\_values(by**=**'Difference', ascending**=False**)

state\_difference.head()

Out[165]:

|  | **State** | **Difference** |
| --- | --- | --- |
| **39** | TX | 9984475.0 |
| **4** | CA | 905885.0 |
| **15** | LA | 634598.0 |
| **7** | FL | 451558.0 |
| **3** | AZ | 441795.0 |

**Step 84: # import** **plotly.graph\_objs as go**

​

**Step 85: # create a bar chart figure**

fig **=** go.Figure(

go.Bar(

x**=**state\_difference.head(10)['State'],

y**=**state\_difference.head(10)['Difference'],

marker\_color**=**'red'

)

)

​

**Step 84:# set figure title and axis labels**

fig.update\_layout(

title\_text**=**'Top 10 States with the Highest Cost of Failures',

xaxis\_title**=**'State',

yaxis\_title**=**'Cost of Failures'

)

​

**Step 85 : *# display the figure***

fig.show()

**Step 86 : # filter the data to exclude Texas**

state\_difference\_filtered **=** state\_difference[state\_difference['State'] **!=** 'TX']

​

**Step 87: # create a bar chart figure**

fig **=** go.Figure(

go.Bar(

x**=**state\_difference\_filtered.head(10)['State'],

y**=**state\_difference\_filtered.head(10)['Difference'],

marker\_color**=**'red'

)

)

​

**Step 88 : # set figure title and axis labels**

fig.update\_layout(

title\_text**=**'Top 10 States (excluding TX) with the Highest Cost of Failures',

xaxis\_title**=**'State',

yaxis\_title**=**'Cost of Failures'

)

​

**Step 89: # display the figure**

fig.show()

**Step 90: import** plotly.graph\_objs **as** go

​

**Step 91: # create a choropleth map figure**

fig **=** go.Figure(data**=**go.Choropleth(

locations**=**state\_difference['State'], *# state codes*

z**=**state\_difference['Difference'], *# difference amount*

locationmode**=**'USA-states', *# set location mode to USA states*

colorscale**=**'YlOrRd', *# set colorscale to yellow-orange-red*

colorbar\_title**=**"Difference" *# set colorbar title*

))

​

**Step 92:** *# set figure title and size*

fig.update\_layout(

title\_text**=**'Cost of Failures per State',

geo**=**dict(scope**=**'usa', projection**=**go.layout.geo.Projection(type**=**'albers usa')),

width**=**800,

height**=**500,

legend**=**dict(

title**=**"Difference",

title\_font**=**dict(size**=**18),

font**=**dict(size**=**14)

)

)

​

*# display the figure*

fig.show()

## Analysis

-Almost all major cities, as major banking areas, are deeply effected by large economic downturns -Single economic downturns cause

In [228]:



*# Reorder the columns of the dataframe*

df **=** df[['Bank Name', 'ID', 'State', 'Failure Date', 'Failure Year', 'Assets', 'Deposits']**+** [c **for** c **in** df.columns **if** c **not** **in** ['Bank Name', 'ID', 'State', 'Failure Date', 'Failure Year', 'Assets', 'Deposits']]]

​

*# Group the data by year and count the number of banks that failed each year*

failures\_per\_year **=** df.groupby('Failure Year').size().reset\_index(name**=**'Num Failures')

​

*# Calculate year-over-year changes and store them in a list*

yearly\_changes **=** [failures\_per\_year['Num Failures'].iloc[i] **-** failures\_per\_year['Num Failures'].iloc[i**-**1] **for** i **in** range(1, len(failures\_per\_year))]

​

**# Calculate mean and standard deviation of yearly changes**

mean **=** np.mean(yearly\_changes)

std\_dev **=** np.std(yearly\_changes)

​

**# Create a normal distribution with the calculated mean and standard deviation**

x **=** np.linspace(mean **-** 3**\***std\_dev, mean **+** 3**\***std\_dev, 100)

y **=** norm.pdf(x, mean, std\_dev)

​

**# Create a plot of the yearly changes and the normal distribution**

fig, ax **=** plt.subplots()

ax.hist(yearly\_changes, bins**=**50, density**=True**, alpha**=**0.6)

ax.plot(x, y, color**=**'red', linestyle**=**'--', linewidth**=**2)

ax.set\_xlabel('Yearly Changes in Number of Failed Banks')

ax.set\_ylabel('Density')

ax.set\_title('Yearly Changes in Number of Failed Banks\nand Normal Distribution')

plt.show()

Chart, line chart

Description automatically generated

In [230]:



year\_changes **=** []

**for** i **in** range(1, len(failures\_per\_year)):

**if** failures\_per\_year['Num Failures'].iloc[i] **-** failures\_per\_year['Num Failures'].iloc[i**-**1] **>=** 15:

year\_changes.append(failures\_per\_year['Failure Year'].iloc[i])

​

year\_changes

Out[230]:

[1981, 1982, 1985, 1986, 1987, 1988, 1989, 2008, 2009]

In [254]:



*# Create a list of colors for the bars*

colors **=** ['red' **if** year **in** [1981, 1982, 1985, 1986, 1987, 1988, 1989, 2008, 2009] **else** 'blue' **for** year **in** failures\_per\_year.index]

​

*# Create a bar chart of the data*

fig, ax **=** plt.subplots()

ax.bar(failures\_per\_year.index, failures\_per\_year.values, color**=**colors)

​

*# Set the chart title and axis labels*

ax.set\_title('15+ Failed Bank Increase')

ax.set\_xlabel('Year')

ax.set\_ylabel('Number of Failed Banks')

​

*# Set the x-axis tick locations and labels to show only every 10 years, starting at the nearest multiple of 10 greater than or equal to 1950*

start\_year **=** (1950 **//** 10) **\*** 10

ax.set\_xticks(range(start\_year, failures\_per\_year.index[**-**1]**+**1, 10))

​

*# Add grid lines*

ax.grid(**True**, axis**=**'y')

ax.grid(**True**, axis**=**'x')

​

*# Display the chart*

plt.show()

Chart, histogram

Description automatically generated

## Analysis

-Years with a 15+ in FDIC bank failures are indicators of the beginning of potentially longer running economic down-turns