Module 9 Spatial Data Analytics and Visualization

Topics:

- Creating choropleth maps.
- Using Folium for mapping data.
- How to create heatmaps.
- Creating a pivot table with seaborn.

In [1]:
```
import pandas as pd
```

In [2]:
```
# Load data from the us_pop.csv file
df_pop = pd.read_csv('us_pop.csv')

df_pop.head(5)
```

Out[2]:

	state	value
0	AL	4859000
1	AK	738400
2	AZ	6828000
3	AR	2978000
4	CA	39145000

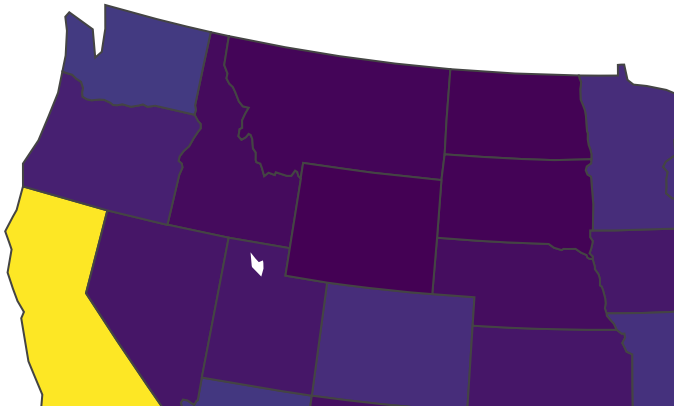
In [3]:
```
import plotly.express as px
```

United States Population Example

In [4]:
```
# Draw a choropleth map.
fig = px.choropleth(df_pop, locations='state', color='value', color_continuous_scal
                    , hover_name='state', locationmode='USA-states', labels={'value
```

In [5]:
```
#Show the chorpleth map.
fig.update_layout(title_text='State Populations', geo_scope='usa')
```

State Populations



United States Area Example

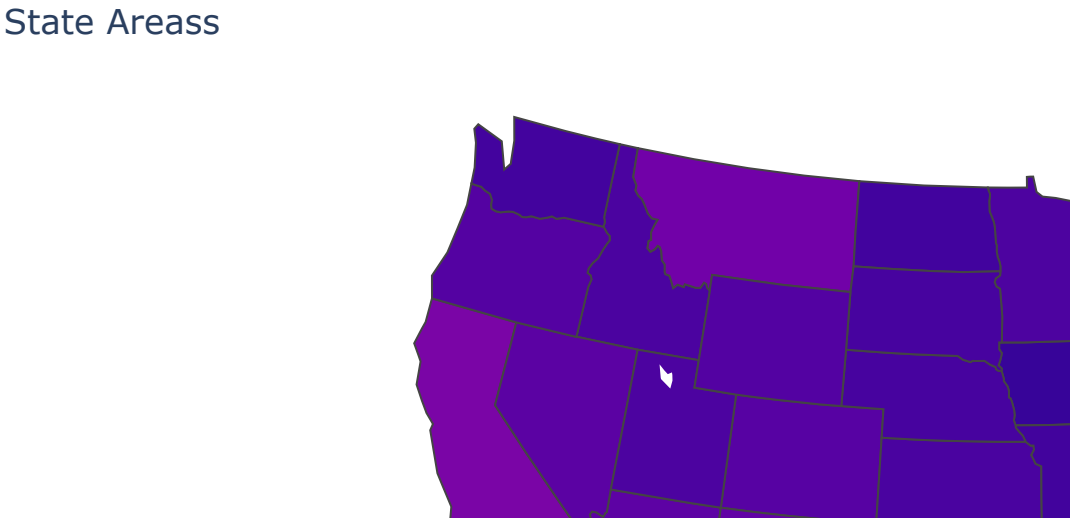
In [6]:
```
# Load data from the us_pop.csv file
df_area = pd.read_csv('us_area.csv')

df_area.head(5)
```

Out[6]:

	state	value
0	AL	52420.0
1	AK	665384.0
2	AZ	113990.0
3	AR	53179.0
4	CA	163695.0

In [7]:
```
# Draw a choropleth map.
fig = px.choropleth(df_area, locations='state', color='value', hover_name='state'
                    , locationmode='USA-states', labels={'value': 'Area'})
```

In [8]:
```
#Show the chorpleth map.
fig.update_layout(title_text='State Areass', geo_scope='usa')
```



COVID Cases World Example

In [9]:
```
# Create a dataframe with COVID information
df_covid = pd.read_csv('covid_19_world.csv')

len(df_covid)
```

Out[9]: 61900

In [10]:
```
df_covid.head()
```

Out[10]:

	dateRep	day	month	year	cases	deaths	countriesAndTerritories	popData2019	cc
0	12/14/2020	14	12	2020	746	6	Afghanistan	38041757.0	
1	12/13/2020	13	12	2020	298	9	Afghanistan	38041757.0	
2	12/12/2020	12	12	2020	113	11	Afghanistan	38041757.0	
3	12/11/2020	11	12	2020	63	10	Afghanistan	38041757.0	
4	12/10/2020	10	12	2020	202	16	Afghanistan	38041757.0	

In [11]:
```
# Using groupby() to group cases by country.
df_covid.groupby('countriesAndTerritories')['cases'].count()
```

Out[11]:
```
countriesAndTerritories
Afghanistan          340
Albania              281
Algeria              345
Andorra              276
Angola               268
                    ...
Wallis_and_Futuna     59
Western_Sahara       233
Yemen                249
Zambia               271
Zimbabwe             269
Name: cases, Length: 214, dtype: int64
```

In [12]:
```
# Using groupby() to group cases by country.
df_covid.groupby('countriesAndTerritories')['cases'].sum()
```

Out[12]:
```
countriesAndTerritories
Afghanistan          49273
Albania              48530
Algeria              92102
Andorra               7338
Angola               16188
                     ...
Wallis_and_Futuna        3
Western_Sahara         766
Yemen                 2083
Zambia               18274
Zimbabwe             11246
Name: cases, Length: 214, dtype: int64
```

In [13]:
```
# Create a new dataframe with the sum results

s = df_covid.groupby('countriesAndTerritories')['cases'].sum()

df_cases = pd.DataFrame({'Country': s.index, 'cases': s.values})

df_cases.head()
```

Out[13]:

	Country	cases
0	Afghanistan	49273
1	Albania	48530
2	Algeria	92102
3	Andorra	7338
4	Angola	16188

In [14]:
```
# Finding out the ISO names of all the countries.

iso_names = pd.read_csv('country_iso.csv', encoding='latin-1')

country_names = list(iso_names['Country'])
country_name_clean = []

for name in country_names:
    country_name_clean.append(name.strip())

country_iso = list(iso_names['iso_3'])
```

In [15]:
```
# Match the country names in the COVID data.

country_iso_matched = []
country_names_matched = []
country_cases_matched = []

for i in range(len(df_cases)):

    name = df_cases.iloc[i, 0]

    for j in range(len(country_name_clean)):

        if name == country_name_clean[j]:

            print(name)
            print(country_iso[j])
            country_names_matched.append(name)
            country_iso_matched.append(country_iso[j].strip())
            country_cases_matched.append(df_cases.iloc[i, 1])
            print('----------------------')
```

```
Afghanistan
AFG
----------------------
Albania
ALB
----------------------
Algeria
DZA
----------------------
Andorra
AND
----------------------
Angola
AGO
----------------------
Anguilla
AIA
----------------------
Argentina
ARG
----------------------
Armenia
ARM
----------------------
Aruba
ABW
----------------------
Australia
AUS
----------------------
Austria
AUT
----------------------
Azerbaijan
AZE
----------------------
Bahrain
BHR
----------------------
Bangladesh
BGD
----------------------
Barbados
BRB
----------------------
Belarus
BLR
----------------------
Belgium
BEL
----------------------
Belize
BLZ
----------------------
Benin
BEN
```

```
----------------------
Bermuda
BMU
----------------------
Bhutan
BTN
----------------------
Botswana
BWA
----------------------
Brazil
BRA
----------------------
Bulgaria
BGR
----------------------
Burundi
BDI
----------------------
Cambodia
KHM
----------------------
Cameroon
CMR
----------------------
Canada
CAN
----------------------
Chad
TCD
----------------------
Chile
CHL
----------------------
China
CHN
----------------------
Colombia
COL
----------------------
Croatia
HRV
----------------------
Cuba
CUB
----------------------
Curaçao
CUW
----------------------
Cyprus
CYP
----------------------
Czechia
CZE
----------------------
Denmark
```

```
DNK
----------------------
Djibouti
DJI
----------------------
Dominica
DMA
----------------------
Ecuador
ECU
----------------------
Egypt
EGY
----------------------
Eritrea
ERI
----------------------
Estonia
EST
----------------------
Eswatini
SWZ
----------------------
Ethiopia
ETH
----------------------
Fiji
FJI
----------------------
Finland
FIN
----------------------
France
FRA
----------------------
Gabon
GAB
----------------------
Georgia
GEO
----------------------
Germany
DEU
----------------------
Ghana
GHA
----------------------
Gibraltar
GIB
----------------------
Greece
GRC
----------------------
Greenland
GRL
----------------------
```

```
Grenada
GRD
----------------------
Guam
GUM
----------------------
Guatemala
GTM
----------------------
Guernsey
GGY
----------------------
Guinea
GIN
----------------------
Guyana
GUY
----------------------
Haiti
HTI
----------------------
Honduras
HND
----------------------
Hungary
HUN
----------------------
Iceland
ISL
----------------------
India
IND
----------------------
Indonesia
IDN
----------------------
Iraq
IRQ
----------------------
Ireland
IRL
----------------------
Israel
ISR
----------------------
Italy
ITA
----------------------
Jamaica
JAM
----------------------
Japan
JPN
----------------------
Jersey
JEY
```

```
----------------------
Jordan
JOR
----------------------
Kazakhstan
KAZ
----------------------
Kenya
KEN
----------------------
Kuwait
KWT
----------------------
Kyrgyzstan
KGZ
----------------------
Latvia
LVA
----------------------
Lebanon
LBN
----------------------
Lesotho
LSO
----------------------
Liberia
LBR
----------------------
Libya
LBY
----------------------
Liechtenstein
LIE
----------------------
Lithuania
LTU
----------------------
Luxembourg
LUX
----------------------
Madagascar
MDG
----------------------
Malawi
MWI
----------------------
Malaysia
MYS
----------------------
Maldives
MDV
----------------------
Mali
MLI
----------------------
Malta
```

```
MLT
----------------------
Mauritania
MRT
----------------------
Mauritius
MUS
----------------------
Mexico
MEX
----------------------
Monaco
MCO
----------------------
Mongolia
MNG
----------------------
Montenegro
MNE
----------------------
Montserrat
MSR
----------------------
Morocco
MAR
----------------------
Mozambique
MOZ
----------------------
Myanmar
MMR
----------------------
Namibia
NAM
----------------------
Nepal
NPL
----------------------
Nicaragua
NIC
----------------------
Nigeria
NGA
----------------------
Norway
NOR
----------------------
Oman
OMN
----------------------
Pakistan
PAK
----------------------
Panama
PAN
----------------------
```

```
Paraguay
PRY
----------------------
Peru
PER
----------------------
Poland
POL
----------------------
Portugal
PRT
----------------------
Qatar
QAT
----------------------
Romania
ROU
----------------------
Rwanda
RWA
----------------------
Senegal
SEN
----------------------
Serbia
SRB
----------------------
Seychelles
SYC
----------------------
Singapore
SGP
----------------------
Slovakia
SVK
----------------------
Slovenia
SVN
----------------------
Somalia
SOM
----------------------
Spain
ESP
----------------------
Suriname
SUR
----------------------
Sweden
SWE
----------------------
Switzerland
CHE
----------------------
Tajikistan
TJK
```

```
----------------------
Thailand
THA
----------------------
Togo
TGO
----------------------
Tunisia
TUN
----------------------
Turkey
TUR
----------------------
Uganda
UGA
----------------------
Ukraine
UKR
----------------------
Uruguay
URY
----------------------
Uzbekistan
UZB
----------------------
Vanuatu
VUT
----------------------
Yemen
YEM
----------------------
Zambia
ZMB
----------------------
Zimbabwe
ZWE
----------------------
```

In [16]:
```
# Create a new dataframe with ISO country codes.
dict_new = {'country': country_names_matched, 'iso': country_iso_matched, 'cases':

df_final = pd.DataFrame(dict_new)
```

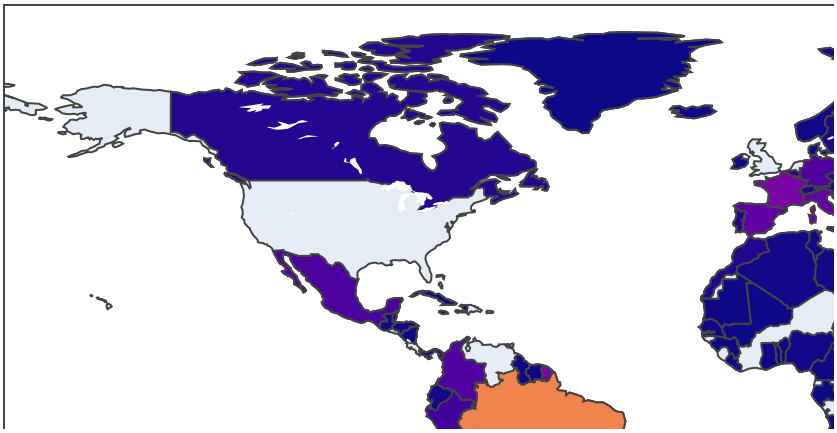
In [17]:
```
df_final
```

Out[17]:

	country	iso	cases
0	Afghanistan	AFG	49273
1	Albania	ALB	48530
2	Algeria	DZA	92102
3	Andorra	AND	7338
4	Angola	AGO	16188
...	...	...	...
138	Uzbekistan	UZB	75094
139	Vanuatu	VUT	1
140	Yemen	YEM	2083
141	Zambia	ZMB	18274
142	Zimbabwe	ZWE	11246

143 rows × 3 columns

In [18]:
```
# Draw a choropleth map.
fig = px.choropleth(df_final, locations='iso', color='cases'
                    , hover_name='country'
                    , color_continuous_scale=px.colors.sequential.Plasma)

# Show the figure
fig.show()
```

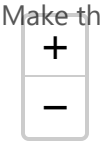


Using Folium to Create a Map

In [19]:
```
## Using Folium for mapping data.
import folium
```

In [20]:
```
# Center the map location using Indianaoplis
indy = folium.Map(location=[39.79, -86.148], zoom_start=7)

# Show the map
indy
```

Out[20]: Make this Notebook Trusted to load map: File -> Trust Notebook

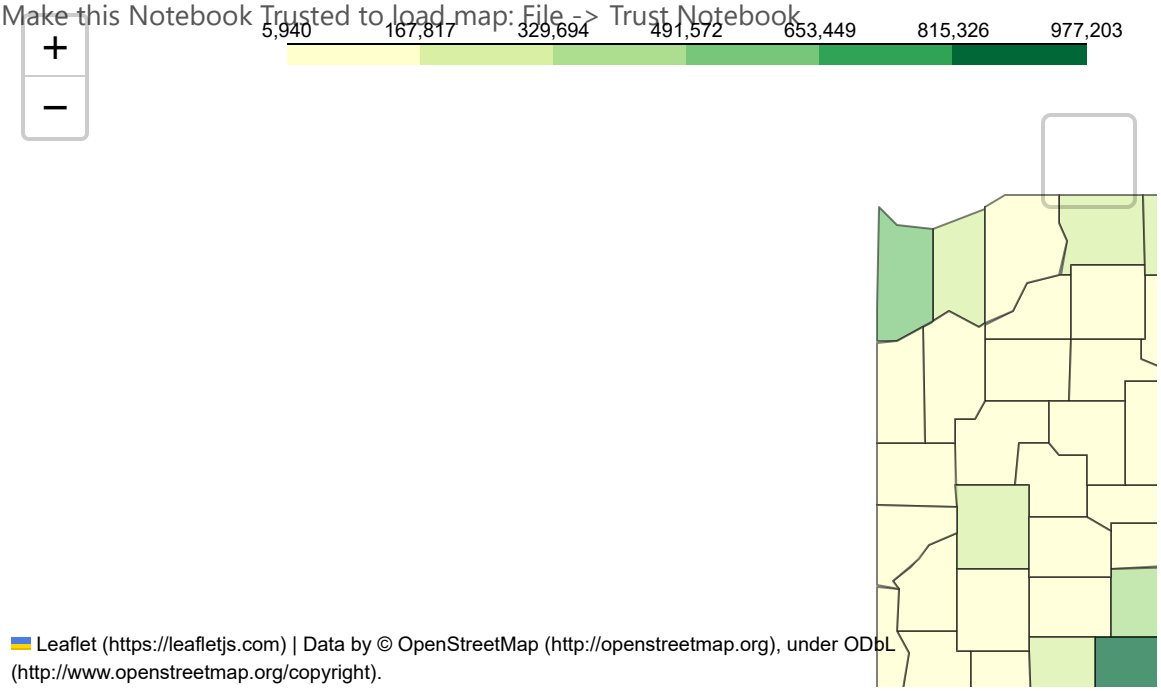


Leaflet (https://leafletjs.com) | Data by © OpenStreetMap (http://openstreetmap.org), under ODbL (http://www.openstreetmap.org/copyright).

In [21]:
```
import folium
#import pandas as pd
```

In [22]:
```
# Using Indiana counties and the unemployment data.

indy = indy = folium.Map(location=[39.79, -86.148], zoom_start=7)

state_geo = 'indiana_counties.json'
state_unemployment = 'indiana.csv'
state_data = pd.read_csv(state_unemployment)

# Create a Folium choropleth
folium.Choropleth(geo_data=state_geo, name='choropleth', data=state_data, columns=[
                  key_on='feature.id', fill_color='YlGn', fill_opacity=0.7, line_op

#
folium.LayerControl().add_to(indy)

# Show the map
indy
```

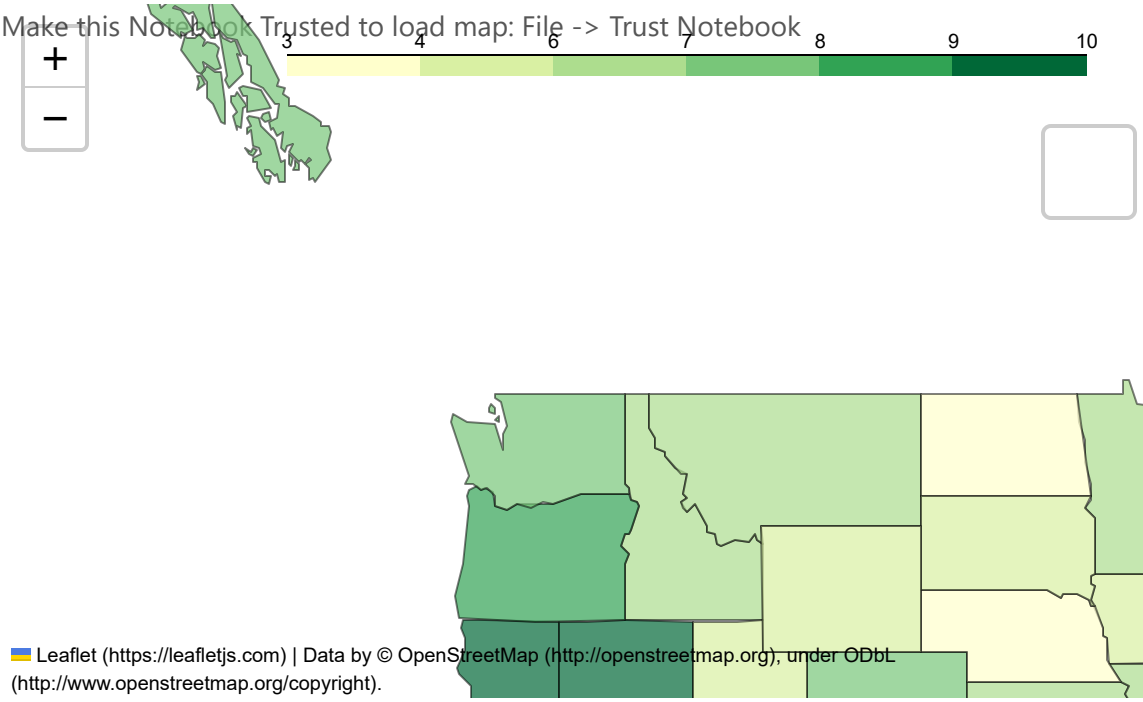
Out[22]: Make this Notebook Trusted to load map: File -> Trust Notebook



In [23]:
```
#Map Create of the whole US using unemployment data
#import folium
#import pandas as pd
```

In [24]:
```
usa = folium.Map(location=[40, -95], zoom_start=4)

state_geo = 'usa_states.json'
state_unemployment = 'us_unemployment_oct2012.csv'

state_data = pd.read_csv(state_unemployment)

# Create the map
folium.Choropleth(geo_data=state_geo, name='choropleth', data=state_data, columns=[
                  key_on='feature.id', fill_color='YlGn', fill_opacity=0.7, line_op

folium.LayerControl().add_to(usa)

# Show the map
usa
```

Out[24]:



Example using California housing Data

- Show the housing data over a local map.

In [25]:
```
#import pandas as pd
#import folium
```

In [26]:
```
df = pd.read_csv('housing.csv')

print(df.head())

print(len(df))
```

```
   longitude  latitude  housing_median_age  total_rooms  total_bedrooms  \
0    -122.23     37.88                41.0        880.0           129.0
1    -122.22     37.86                21.0       7099.0          1106.0
2    -122.24     37.85                52.0       1467.0           190.0
3    -122.25     37.85                52.0       1274.0           235.0
4    -122.25     37.85                52.0       1627.0           280.0

   population  households  median_income  median_house_value ocean_proximity
0       322.0       126.0         8.3252            452600.0        NEAR BAY
1      2401.0      1138.0         8.3014            358500.0        NEAR BAY
2       496.0       177.0         7.2574            352100.0        NEAR BAY
3       558.0       219.0         5.6431            341300.0        NEAR BAY
4       565.0       259.0         3.8462            342200.0        NEAR BAY
20640
```

In [27]:
```
# Drop the duplicates from the dataframe
df2 = df.copy().drop_duplicates(['longitude', 'latitude'])[0:100]

print(len(df2))
```

```
100
```

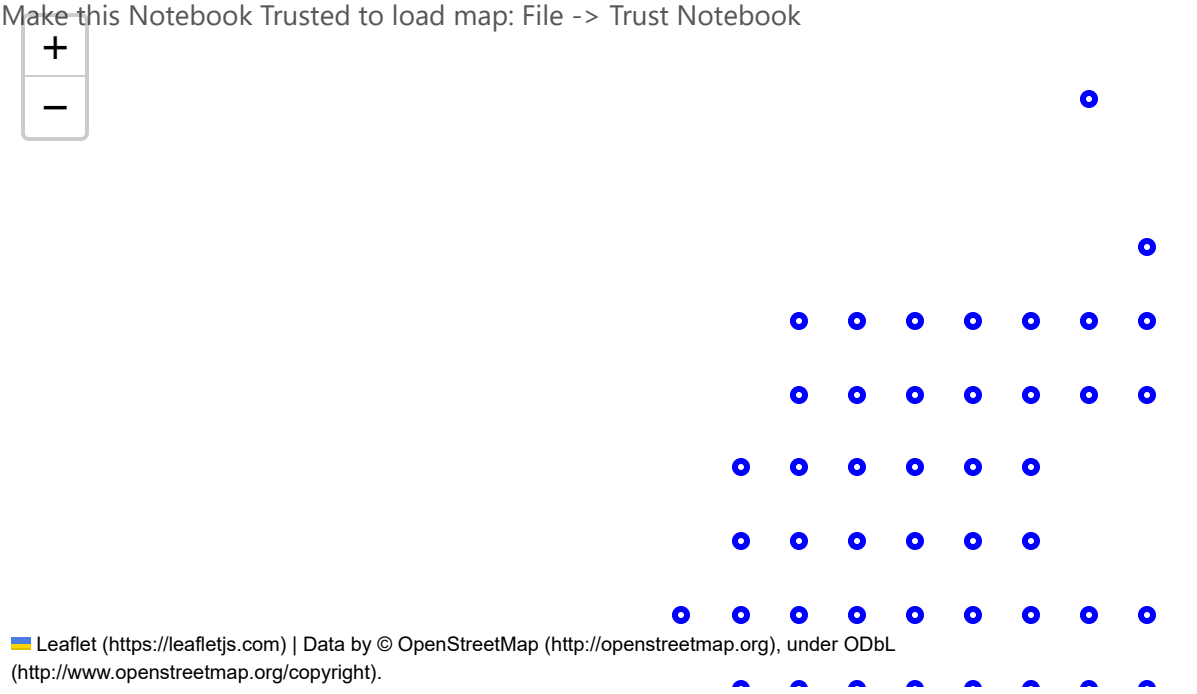
In [28]:
```
df2.head()
```

Out[28]:

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population	hou
0	-122.23	37.88	41.0	880.0	129.0	322.0	
1	-122.22	37.86	21.0	7099.0	1106.0	2401.0	
2	-122.24	37.85	52.0	1467.0	190.0	496.0	
3	-122.25	37.85	52.0	1274.0	235.0	558.0	
6	-122.25	37.84	52.0	2535.0	489.0	1094.0	

In [29]:
```
housing_map = folium.Map(location=[df2.latitude.mean(), df2.longitude.mean()], zoom

for coord in df2[['latitude','longitude']].values:
    folium.CircleMarker(location=[coord[0], coord[1]], radius=3, color='blue').add_

# Show the map
housing_map
```

Out[29]:
Make this Notebook Trusted to load map: File -> Trust Notebook



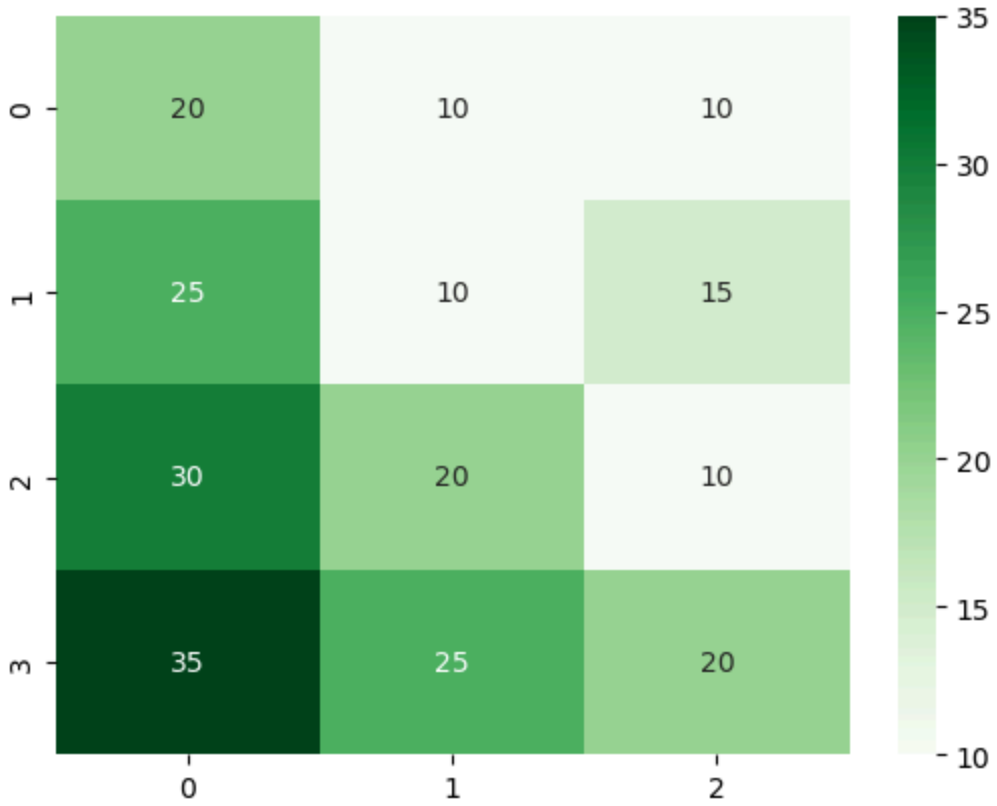
Creating heatmaps

In [30]:
```
import seaborn as sns
```

In [31]:
```
# Create a list
l = [[20, 10, 10],[25, 10, 15],[30, 20, 10],[35, 25, 20]]
```

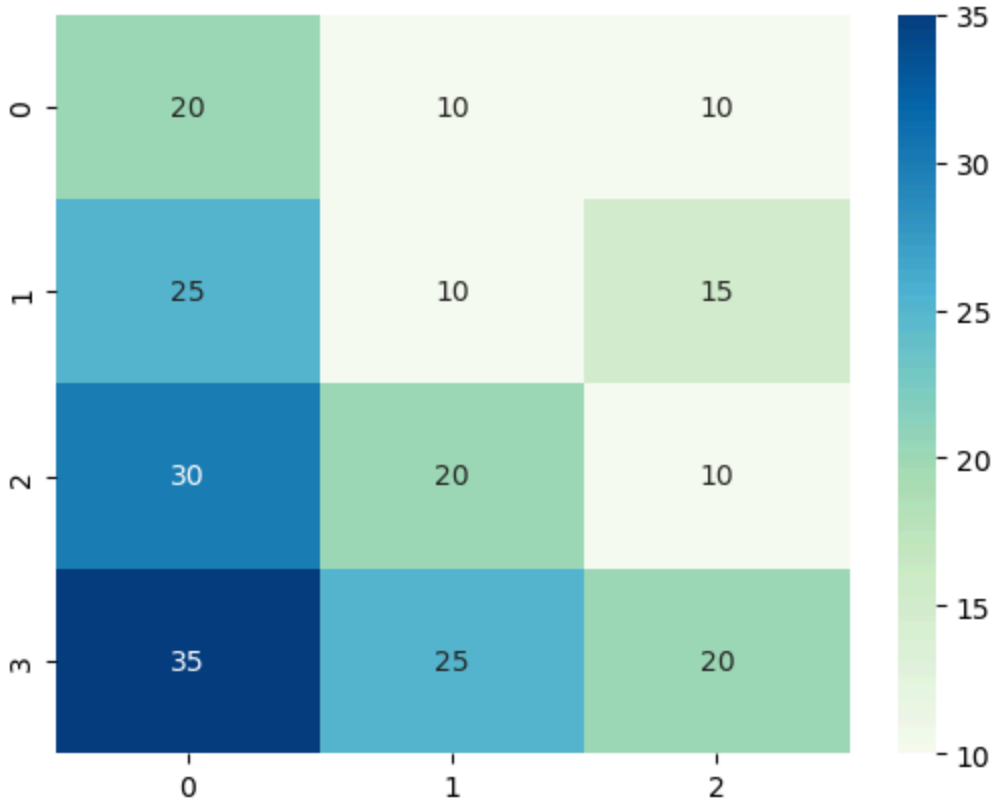
In [32]:
```
# Create a heatmap
sns.heatmap(l, cmap='Greens', annot=True)
```

Out[32]: <Axes: >



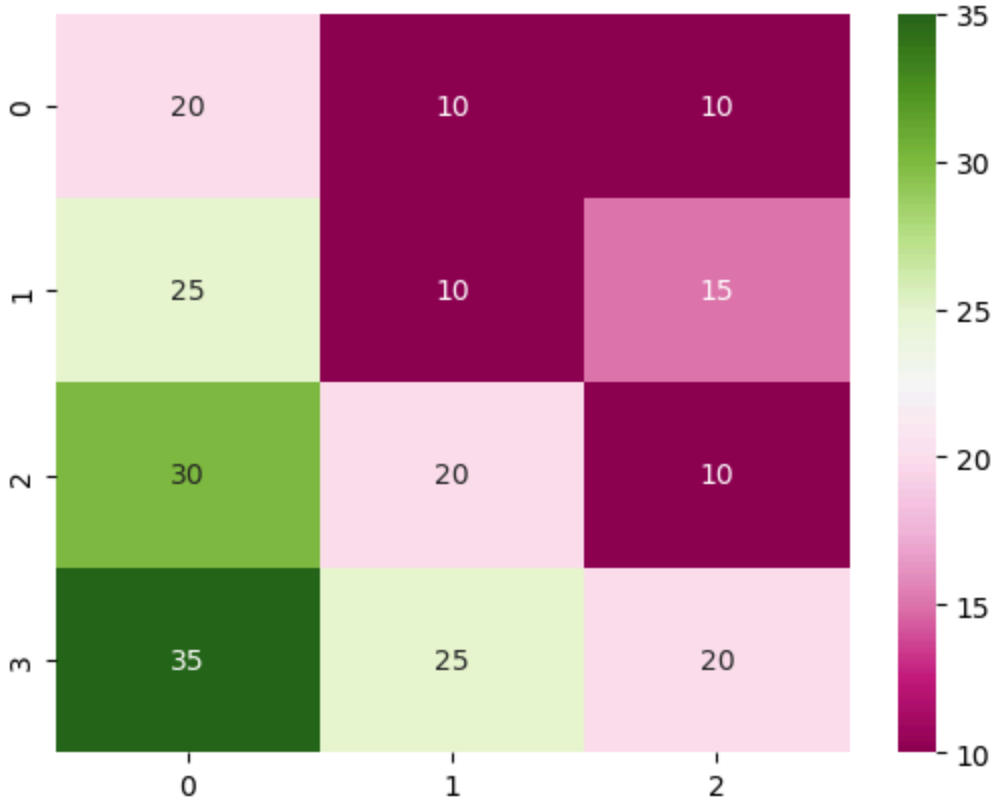
In [33]:
```
# Create a heatmap using two colors
sns.heatmap(l, cmap='GnBu', annot=True)
```

Out[33]: <Axes: >



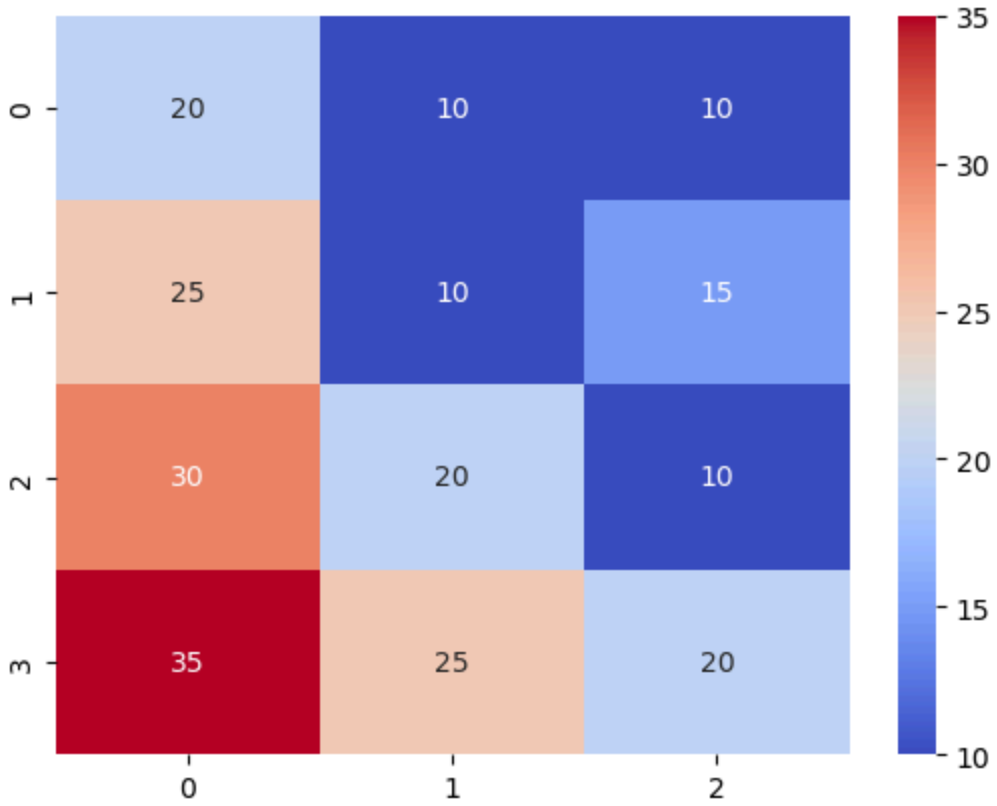
In [34]:
```
# Create a heatmap using three colors
sns.heatmap(l, cmap='PiYG', annot=True)
```

Out[34]: <Axes: >



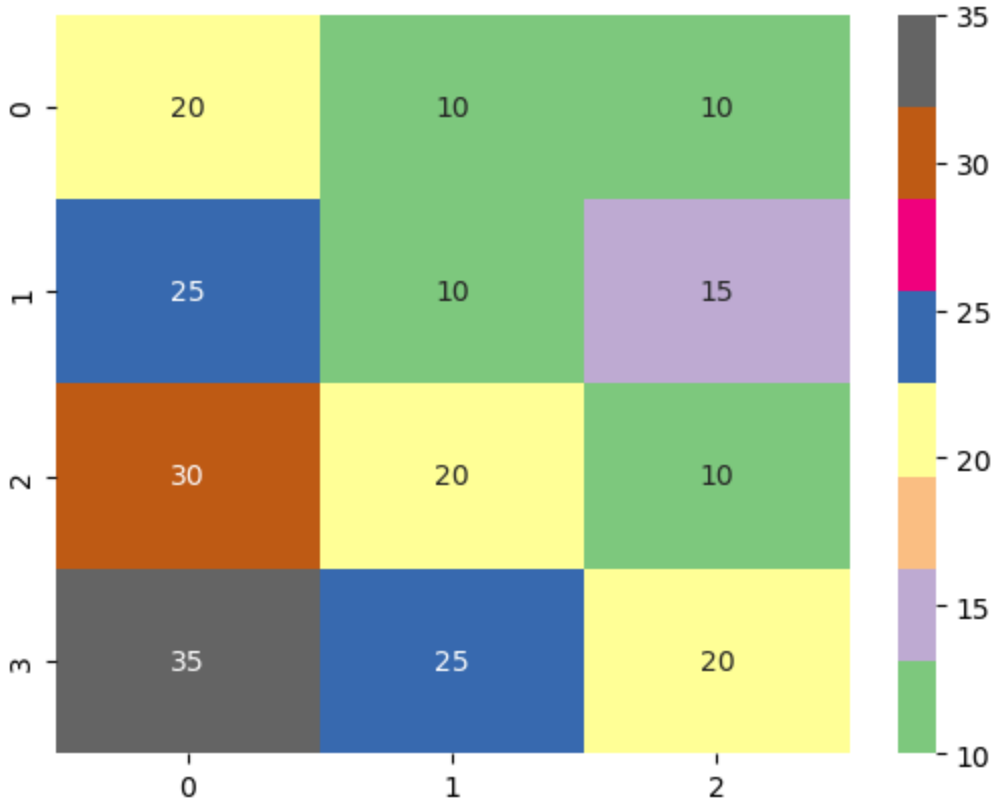
In [35]:
```
# Create a heatmap using predefined color palette.
sns.heatmap(l, cmap='coolwarm', annot=True)
```

Out[35]: <Axes: >



In [36]:
```
# Create a heatmap that is qualitative.
sns.heatmap(l, cmap='Accent', annot=True)
```

Out[36]: <Axes: >



Creating a pivot table with seaborn.

- uses a seaborn dataset
- and group data

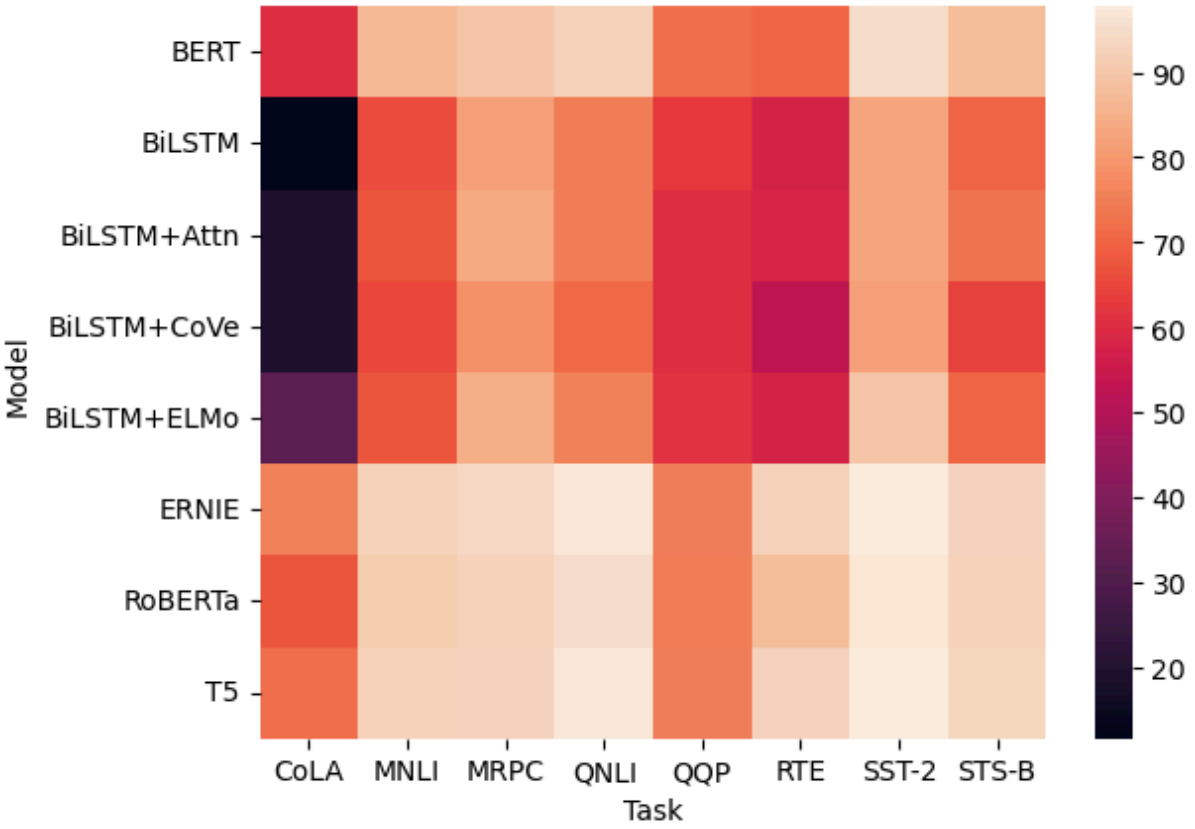
In [37]:
```
glue = sns.load_dataset('glue').pivot(index='Model', columns='Task', values='Score'

glue
```

Out[37]:

Task	CoLA	MNLI	MRPC	QNLI	QQP	RTE	SST-2	STS-B
Model								
BERT	60.5	86.7	89.3	92.7	72.1	70.1	94.9	87.6
BiLSTM	11.6	65.6	81.8	74.6	62.5	57.4	82.8	70.3
BiLSTM+Attn	18.6	67.6	83.9	74.3	60.1	58.4	83.0	72.8
BiLSTM+CoVe	18.5	65.4	78.7	70.8	60.6	52.7	81.9	64.4
BiLSTM+ELMo	32.1	67.2	84.7	75.5	61.1	57.4	89.3	70.3
ERNIE	75.5	92.3	93.9	97.3	75.2	92.6	97.8	93.0
RoBERTa	67.8	90.8	92.3	95.4	74.3	88.2	96.7	92.2
T5	71.6	92.2	92.8	96.9	75.1	92.8	97.5	93.1

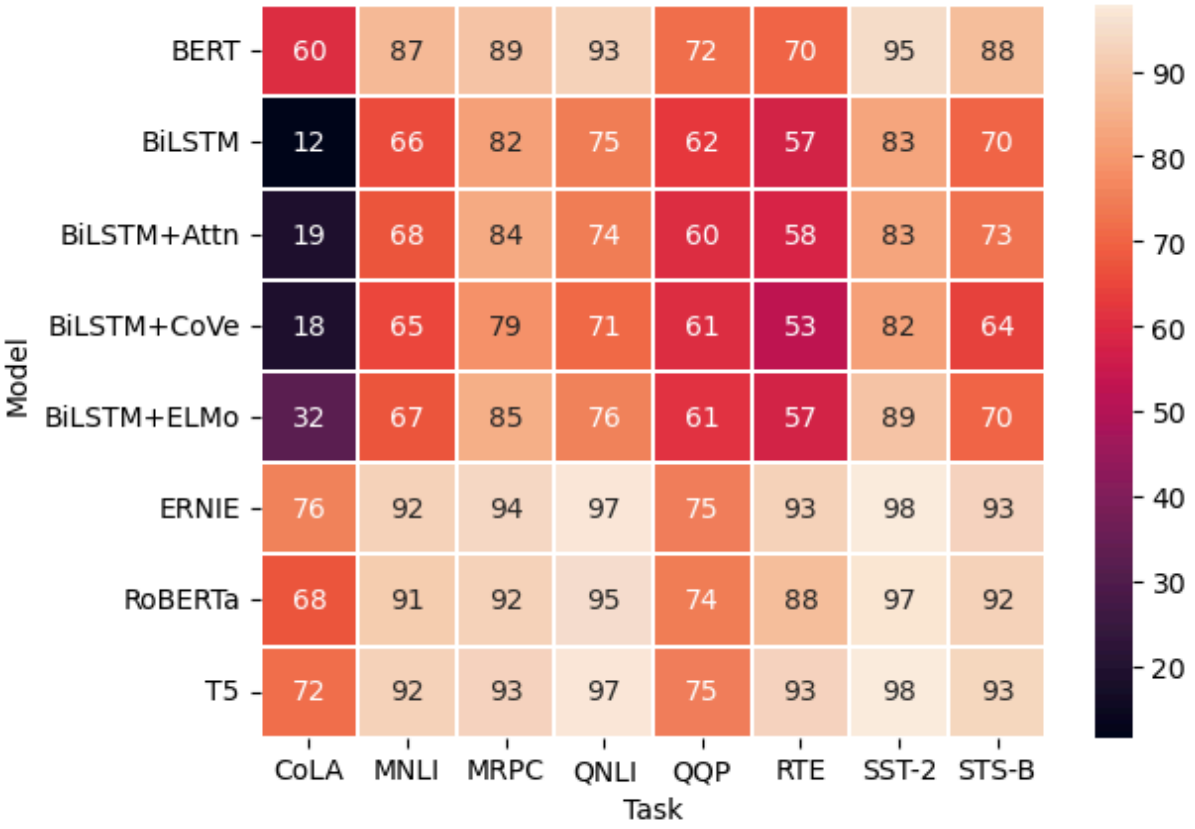
In [38]:
```
# Create a heatmap
sns.heatmap(glue)
```

Out[38]:
```
<Axes: xlabel='Task', ylabel='Model'>
```



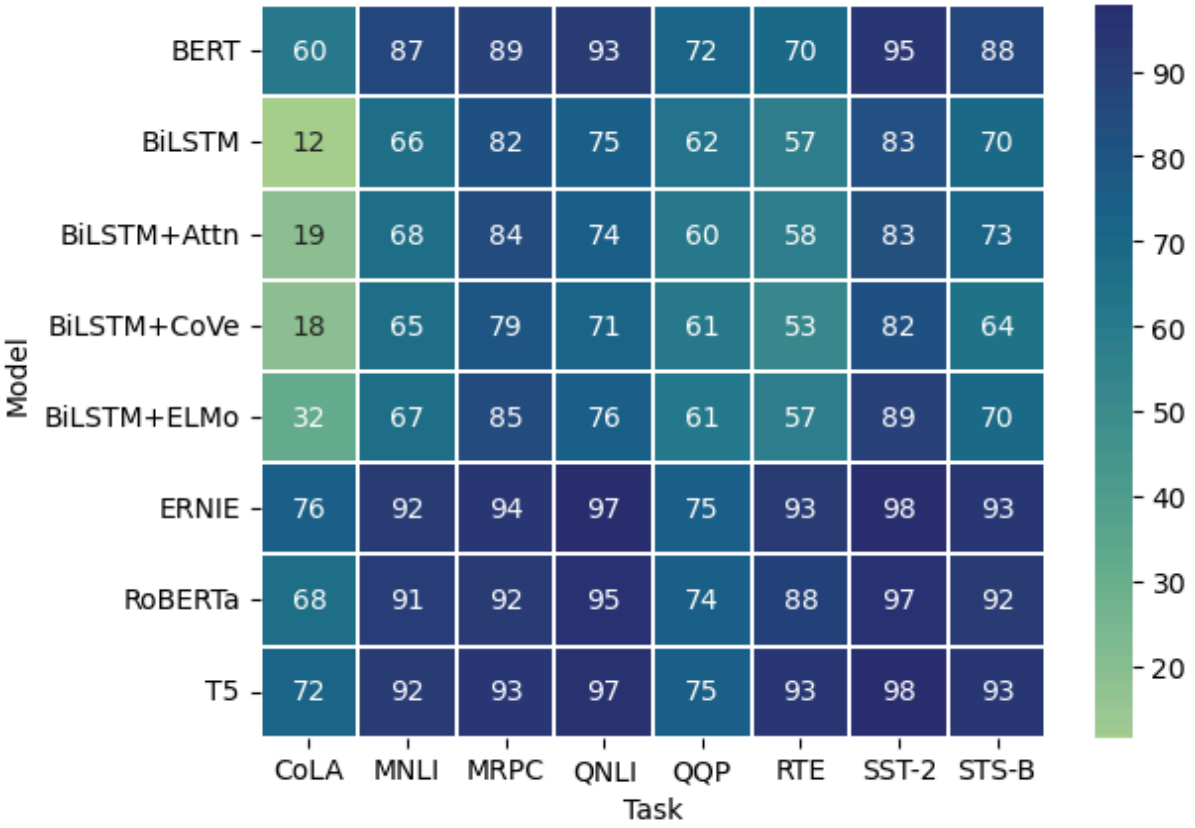
In [39]:
```
sns.heatmap(glue, annot=True, linewidth=0.3)
```

Out[39]:
```
<Axes: xlabel='Task', ylabel='Model'>
```



In [40]:
```
# Using different color map.
sns.heatmap(glue, annot=True, linewidth=0.3, cmap='crest')
```

Out[40]: <Axes: xlabel='Task', ylabel='Model'>



In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]: