U.S. Migration and Machine Learning

Azzy Caceres, Michael Striffler, Kholiswa Tsotetsi

Introduction

Utilizing data from the U.S. Census Bureau American Community Survey & Zillow Research data hub, we sought to use machine learning techniques to see if we could create a predictive model that would accurately calculate the potential migration in or out of a state in the future. Our model uses a few variables that we thought might influence migration, including; average home price, unemployment rate, and median income. While we tested multiple models , linear regression gave us the best R score of nearly 87%.

Our Hypothesis

The idea of migration peaked our interest as there have been many changes throughout the recent years that we felt could have an impact on why people are moving. In 2008 we saw the market crash, leading to years of low housing market, we have seen state and local taxes be capped on a tax level starting in 2018, and most recently, we have seen a pandemic that has created increased awareness for people that live in certain areas. Suddenly people have the freedom to work for anywhere in the world remotely.

- ★ **Are people looking at the home prices in the states they are looking to migrate to?**
- ★ **Do the median income and unemployment rate of a state sway someone to move (or stay away from a state)?**
- ★ **Did the cap on property taxes cause people to move to areas where the property taxes are lower?**
- ★ **What were the states that people ended up migrating to?**

Data Cleaning

Utilized Excel and Python Pandas library

```
In [1]: import pandas as pd
        from sqlalchemy import create_engine
        from sql_keys import username, password
```

```
In [2]: migration2010 = pd.read_excel('cleaned excels/state_to_state_migrations_table_2010_cleaned2.xls')
        migration2011 = pd.read_excel('cleaned excels/state_to_state_migrations_table_2011_cleaned2.xls')
        migration2012 = pd.read_excel('cleaned excels/state_to_state_migrations_table_2012_cleaned2.xls')
        migration2013 = pd.read_excel('cleaned excels/State_to_State_Migrations_Table_2013_clean2.xls')
        migration2014 = pd.read_excel('cleaned excels/State_to_State_Migrations_Table_2014_clean2.xls')
        migration2015 = pd.read_excel('cleaned excels/State_to_State_Migrations_Table_2015_clean2.xls')
        migration2016 = pd.read_excel('cleaned excels/State_to_State_Migrations_Table_2016_clean2.xls')
        migration2017 = pd.read_excel('cleaned excels/State_to_State_Migrations_Table_2017_clean2.xls')
        migration2018 = pd.read_excel('cleaned excels/State_to_State_Migrations_Table_2018_clean2.xls')
        migration2019 = pd.read_excel('cleaned excels/State_to_State_Migrations_Table_2019_clean2.xls')
```

```
In [3]: migration2019.head()
```

Out[3]:

	Year	Before Tax Change	Current residence in	Total Population	Same house 1 year ago	Same state of residence 1 year ago	Total_Incoming	Total_Outgoing	Net Migration	Net Migrationpn	...	Utah	Vermont	Virginia	Washi
0	2019	no	Alabama	4849509	4197698	534842	104780	98704	6076	positive	...	2083.0	0.0	2876.0	1
1	2019	no	Alaska	722063	609362	74375	34031	50134	-16103	negative	...	1092.0	44.0	2216.0	1
2	2019	no	Arizona	7200620	6038776	857935	253295	173631	79664	positive	...	8917.0	662.0	3837.0	13
3	2019	no	Arkansas	2989054	2559444	361762	59723	64524	-4801	negative	...	710.0	0.0	671.0	
4	2019	no	California	39064048	34394220	3943139	460204	633551	-173347	negative	...	8504.0	784.0	24506.0	31

5 rows × 67 columns

```
In [4]: combinedyears = migration2010.append([migration2011, migration2012,migration2013, migration2014,migration2015,
        combinedyears
```

Out[4]:

	Year	Before Tax Change	Current residence in	Total Population	Same house 1 year ago	Same state of residence 1 year ago	Total_Incoming	Total_Outgoing	Net Migration	Net Migrationpn	...	Utah	Vermont	Virginia	Wi
0	2010	yes	Alabama	4729509	3987155	620465	108723	99850	8873	positive	...	1336.0	0.0	2490.0	
1	2010	yes	Alaska	702974	565031	95878	36326	94848	-58522	negative	...	1274.0	353.0	714.0	
2	2010	yes	Arizona	6332786	5069002	1001991	222725	177056	45669	positive	...	7164.0	664.0	3413.0	
3	2010	yes	Arkansas	2888304	2357806	412997	79127	64264	14863	positive	...	361.0	0.0	494.0	
4	2010	yes	California	36907897	30790221	5413287	444749	575165	-130416	negative	...	10653.0	525.0	14232.0	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
46	2019	no	Virginia	8439982	7201805	903735	264855	276849	-11994	negative	...	1736.0	1633.0	NaN	
47	2019	no	Washington	7527366	6253469	977928	231956	199758	32198	positive	...	5440.0	391.0	6656.0	
48	2019	no	West Virginia	1773280	1563611	164739	39548	40460	-912	negative	...	0.0	19.0	6008.0	
49	2019	no	Wisconsin	5760481	5001140	634732	107973	101668	6305	positive	...	808.0	0.0	1878.0	
50	2019	no	Wyoming	572884	473128	68127	30247	23287	6960	positive	...	1746.0	0.0	415.0	

510 rows × 67 columns

```
In [5]: combinedyears = combinedyears.fillna(0)
```

```
In [6]: combinedyears.to_excel("2010-2019combineddata.xls", index= False, header= True)
```

```
In [7]: combinedyears
```

Out[7]:

	Year	Before Tax Change	Current residence in	Total Population	Same house 1 year ago	Same state of residence 1 year ago	Total_Incoming	Total_Outgoing	Net Migration	Net Migrationpn	...	Utah	Vermont	Virginia	Wi
0	2010	yes	Alabama	4729509	3987155	620465	108723	99850	8873	positive	...	1336.0	0.0	2490.0	
1	2010	yes	Alaska	702974	565031	95878	36326	94848	-58522	negative	...	1274.0	353.0	714.0	
2	2010	yes	Arizona	6332786	5069002	1001991	222725	177056	45669	positive	...	7164.0	664.0	3413.0	
3	2010	yes	Arkansas	2888304	2357806	412997	79127	64264	14863	positive	...	361.0	0.0	494.0	
4	2010	yes	California	36907897	30790221	5413287	444749	575165	-130416	negative	...	10653.0	525.0	14232.0	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
46	2019	no	Virginia	8439982	7201805	903735	264855	276849	-11994	negative	...	1736.0	1633.0	0.0	
47	2019	no	Washington	7527366	6253469	977928	231956	199758	32198	positive	...	5440.0	391.0	6656.0	
48	2019	no	West Virginia	1773280	1563611	164739	39548	40460	-912	negative	...	0.0	19.0	6008.0	
49	2019	no	Wisconsin	5760481	5001140	634732	107973	101668	6305	positive	...	808.0	0.0	1878.0	
50	2019	no	Wyoming	572884	473128	68127	30247	23287	6960	positive	...	1746.0	0.0	415.0	

510 rows × 67 columns

```
In [8]: #rds_connection_string = "postgres:bootcamp@localhost:5432/satellite"
        #insert password>@localhost:5432/customer_db"
        engine = create_engine(f'postgresql://{username}:{password}@localhost:5432/housingdata')
```

```
In [9]: engine.table_names()
```

Out[9]: ['housingdata']

```
In [10]: combinedyears.to_sql(name='housingdata', con=engine, if_exists='append', index=False)
```

```
In [11]: pd.read_sql_query('select * from housingdata', con=engine).head()
```

Out[11]:

	Year	Before Tax Change	Current residence in	Total Population	Same house 1 year ago	Same state of residence 1 year ago	Total_Incoming	Total_Outgoing	Net Migration	Net Migrationpn	...	Utah	Vermont	Virginia	Washir
0	2010	yes	Alabama	4729509	3987155	620465	108723	99850	8873	positive	...	1336	0	2490	
1	2010	yes	Alaska	702974	565031	95878	36326	94848	-58522	negative	...	1274	353	714	
2	2010	yes	Arizona	6332786	5069002	1001991	222725	177056	45669	positive	...	7164	664	3413	1
3	2010	yes	Arkansas	2888304	2357806	412997	79127	64264	14863	positive	...	361	0	494	
4	2010	yes	California	36907897	30790221	5413287	444749	575165	-130416	negative	...	10653	525	14232	3

5 rows × 67 columns

Machine Learning: Lessons and Limitations

- K means algorithm
- Logistic Regression
- Random Forest Regression
- Linear Regression

Machine Learning: Lessons and Limitations

$$Y_i = \theta_0 + \theta_1 X_{i1} + \theta_2 X_{i2} + \ldots + \theta_p X_{ip}$$

Net Migration

Y-intercept

Selected Factors: Average Median Income, Unemployment Rate, Average Home Price, Total Population, etc

- **Linear Regression**

Linear Regression Model

In [6]:
```
# Assign the data to X and y
# Note: Sklearn requires a two-dimensional array of values
# so we use reshape to create this

X = combined_data[columns].values
y = combined_data["Net Migration"].values.reshape(-1, 1)

print("Shape: ", X.shape, y.shape)
```

```
Shape:  (510, 57) (510, 1)
```

In [ ]:
```

```

In [7]:
```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=42)
```

In [8]:
```
# Create the model

from sklearn.linear_model import LinearRegression

model = LinearRegression()
```

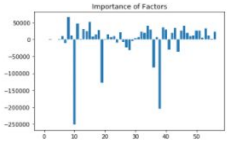
In [9]:
```
# Fit the model to the training data.
model.fit(X_train, y_train)
```

Out[9]:
```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=False)
```

In [10]:
```
#print('Weight coefficients: ', model.coef_)
#print('y-axis intercept: ', model.intercept_)
importance=model.coef_[0]

features_importance = []

for i, v in enumerate(importance):
    features_importance.append(v)


features_importance_names = zip(features_importance, columns)
print(tuple(features_importance_names))
plt.bar([x for x in range(len(importance))], importance)
plt.title("Importance of Factors")
plt.show()
```

```
features_importance_names = zip(features_importance, columns)
print(tuple(features_importance_names))
plt.bar([x for x in range(len(importance))], importance)
plt.title('Importance of Factors')
plt.show()
```

```
((-0.4999779247754093 7, 'Median Income'), (0.009667487406501231, 'Avg Home Price'), (-720.4332840204958, 'Unemployment Rate'), (0.004255962949969216, 'Total Population'), (839.9613214344755, 'Before Tax Change_no'), (-839.9613215548625, 'Before Tax Change_yes'), (9836.869339436816, 'Current residence in Alabama'), (-10643.683745839615, 'Current residence in Alaska'), (6534 8.665553039061, 'Current residence in Arizona'), (11032.867409629249, 'Current residence in Arkansas'), (-250705.572402186 7, 'Current residence in California'), (66897.44829163456, 'Current residence in Colorado'), (-636.8638162265818, 'Current residence in Connecticut'), (30568.486480959495, 'Current residence in Delaware'), (24130.338008643187, 'Current residence in District of Columbia'), (51434.7054079517, 'Current residence in Florida'), (8800.032510250362, 'Current residence in Georgia'), (14367.466206576381, 'Current residence in Hawaii'), (27240.01687424112, 'Current residence in Idaho'), (-127847.42929004526, 'Current residence in Illinois'), (-1995.8214603645 31, 'Current residence in Indiana'), (14213.725537417607, 'Current residence in Iowa'), (7242.557852238726, 'Current residence in Kansas'), (10531.536955421832, 'Current residence in Kentucky'), (-9225.517699604402, 'Current residence in Louisiana'), (20234.787889258630, 'Current residence in Maine'), (-7012.343573689408, 'Current residence in Maryland'), (-23341.672817 30118, 'Current residence in Massachusetts'), (-31851.14782409899, 'Current residence in Michigan'), (-4792.853331435148, 'Current residence in Minnesota'), (3085.145311198521, 'Current residence in Mississippi'), (6226.301201138429, 'Current residence in Missouri'), (22878.2069667634, 'Current residence in Montana'), (39633.659881264823, 'Current residence in Nebraska'), (40476.046087893774, 'Current residence in Nevada'), (29519.915946943212, 'Current residence in New Hampshire'), (-82916.49099203211, 'Current residence in New Jersey'), (6671.671618734348, 'Current residence in New Mexico'), (-204528.866686146585, 'Current residence in New York'), (34997.134211365365, 'Current residence in North Carolina'), (29562.184193876463, 'Current residence in North Dakota'), (-30926.796971926244, 'Current residence in Ohio'), (10972.693489377018, 'Current residence in Oklahoma'), (33972.920465573 65, 'Current residence in Oregon'), (-36062.943728043 65, 'Current residence in Pennsylvania'), (25791.023518009562, 'Current residence in Rhode Island'), (39920.09399505297, 'Current residence in South Carolina'), (18857.549618107 47, 'Current residence in South Dakota'), (10536.35526911296, 'Current residence in Tennessee'), (11485.818135758549, 'Current residence in Texas'), (26184.733693120197, 'Current residence in Utah'), (25453.036228881726, 'Current residence in Vermont'), (4785.482317825285, 'Current residence in Virginia'), (32316.996319451533, 'Current residence in Washington'), (11770.807642578737, 'Current residence in West Virginia'), (1991.7312534510747, 'Current residence in Wisconsin'), (23239.21064812136, 'Current residence in Wyoming'))
```



In [11]:
```
from sklearn.metrics import mean_squared_error, r2_score

# Use our model to make predictions
predicted = model.predict(X_test)

# Score the predictions with mse and r2
mse = mean_squared_error(y_test, predicted)
r2 = r2_score(y_test, predicted)

print(f"Mean Squared Error (MSE): {mse}")
print(f"R-squared (R2 ): {r2}")
```

```
Mean Squared Error (MSE): 290125209.63759637
R-squared (R2 ): 0.8718974740021412
```

In [17]:
```
print('y-axis intercept: ', model.intercept_)
```

```
y-axis intercept:  [4600.38253616]
```

In [12]:
```
model.score(X_test, y_test)
```

Out[12]:
```
0.8718974740021412
```

In [13]:
```
rsquared = 1 - (1-model.score(X_test, y_test))*(len(y)-1)/(len(y)-X.shape[1]-1)
rsquared
```

Out[13]:
```
0.8557429519103405
```

In [14]:
```
# Note: we have to transform our min and max values
# This is the required format for `model.predict()`

x_min = np.array([[X.min()]])
x_max = np.array([[X.max()]])
print(f"Min X Value: {x_min}")
print(f"Max X Value: {x_max}")
```

```
Min X Value: [[0.]]
Max X Value: [[39114089.]]
```

In [15]:
```
from sklearn.metrics import mean_squared_error, r2_score

# Use our model to make predictions
predicted = model.predict(X_test)

# Score the predictions with mse and r2
mse = mean_squared_error(y_test, predicted)
r2 = r2_score(y_test, predicted)

print(f"Mean Squared Error (MSE): {mse}")
print(f"R-squared (R2 ): {r2}")
```

```
Mean Squared Error (MSE): 290125209.63759637
R-squared (R2 ): 0.8718974740021412
```

In [16]:
```
model.score(X_test, y_test)
```

Out[16]:
```
0.8718974740021412
```

In [ ]:
```

```

Deployment of Model

```
9
10  app = Flask(__name__)
11
12
13  # Load the model
14
15  with open('predict.pkl', 'rb') as file:
16      testrun = pickle.load(file)
17
18  #testrun = pickle.load(open('model.pkl','rb'))
19  # testrun = load_model("migration_trained.h5")
20
21  ##Define app routoe
22  @app.route('/', methods=['GET', 'POST'])
23
24
25
26  def index():
27      if request.method == 'POST':
28
29
30
31          medincome = (request.form['Median Income'])
32          homeprice = request.form.get('Avg Home Price')
33          umemployment = request.form.get("Unemployment Rate")
34          totalpop = request.form.get("Total Population")
35          usState = request.form.get("State")
36
37
38
```

Flask App

```
        predMigrate = testrun.predict(x2)

        return render_template('tableau.html', state=usState , predMigrate= predMigrate)

    return render_template('tableau.html')
```

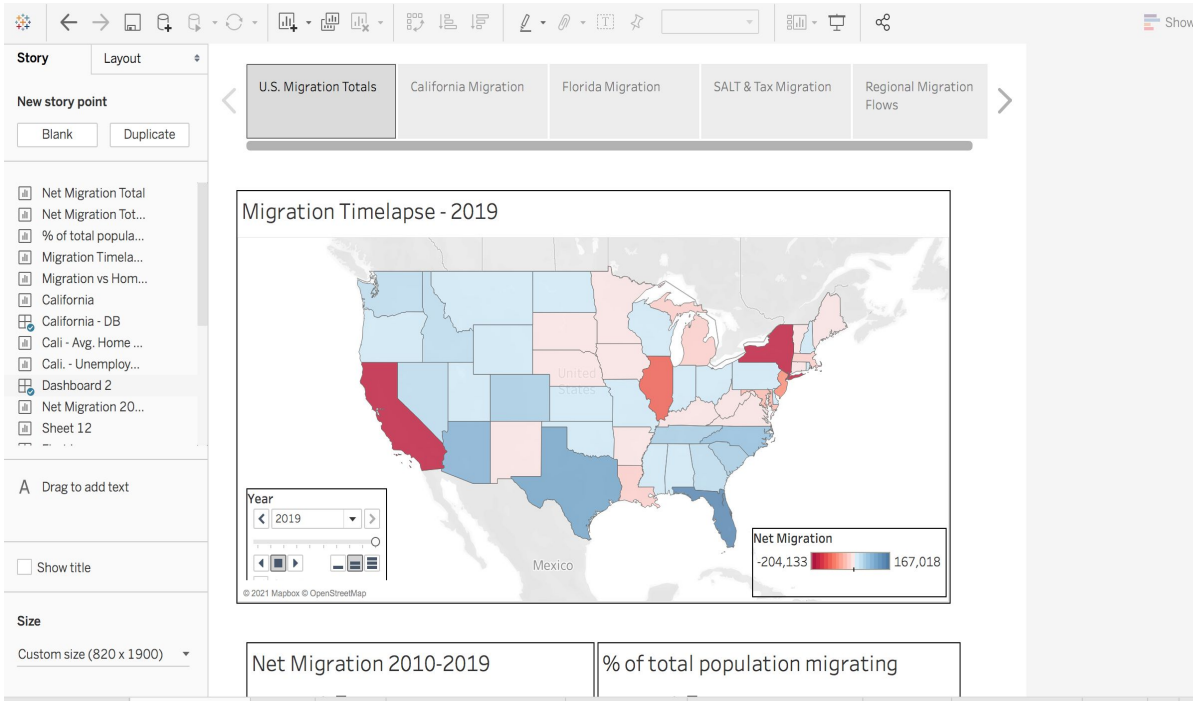
HTML Page

```
        </form>
        State: {{state}}
        <br>
        Migration Prediction : {{predMigrate[0]}}
        </p>
```

Tableau Data Visualizations

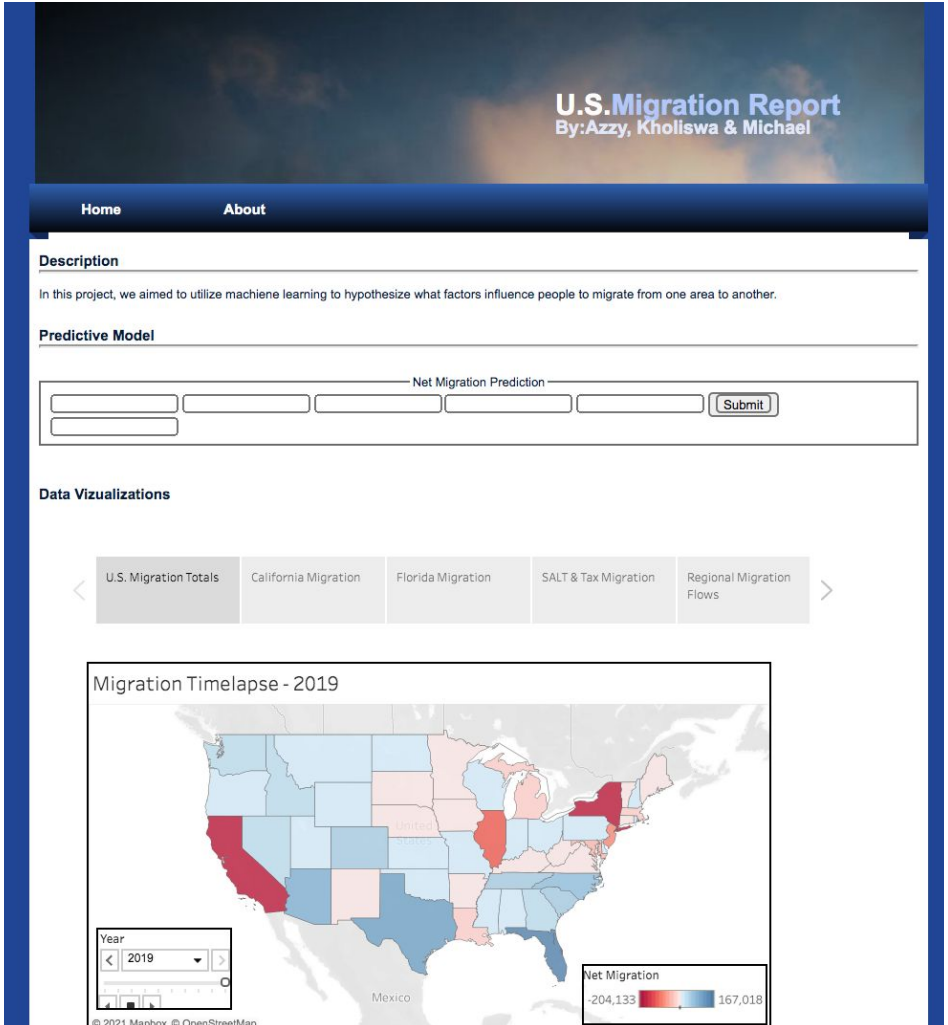
We saved our combined data file that we cleaned and imported it into Tableau to create our visualizations.

We saved them to a ‘story’, published it to Tableau Public and used the embed code to display it on the site.



Website/HTML

Ultimately, we combined our linear regression model, flask app & HTML to create a functional website where users can input data and run our model. The output returns net migration prediction for the specified state.



Thank you!