ECE 356 Project: 2020 Election

Group 1

Members

Aarti Vasudevan, ID: 20746541
Aishwarya Srinivas, ID: 20720876
Indraj Kang, ID: 20628685

Responsibilities

Aarti Vasudevan - frontend, testing
Aish Srinivas - creating and normalizing database, building queries, frontend, testing, created diagrams
Indraj Kang - frontend, testing

Scope

This is a CLI program where the main audience is political parties who want to know about the voting results and demographics of a particular county/state. This is the following info they can know about:

- Total votes and which party won by how many votes in a county or state
- Get the most popular tweets from #TweetsTrump and #TweetsBiden and check how many likes and retweets they got
- View the demographics about a county or state
- Get breakdown of votes per party for each county/state
- Add comments about a county or state
- Data Mining - Find out what are the most important demographics for determining a Trump vs Biden win for a particular county.

This project was done using Python and MySQL on the project_1 database.

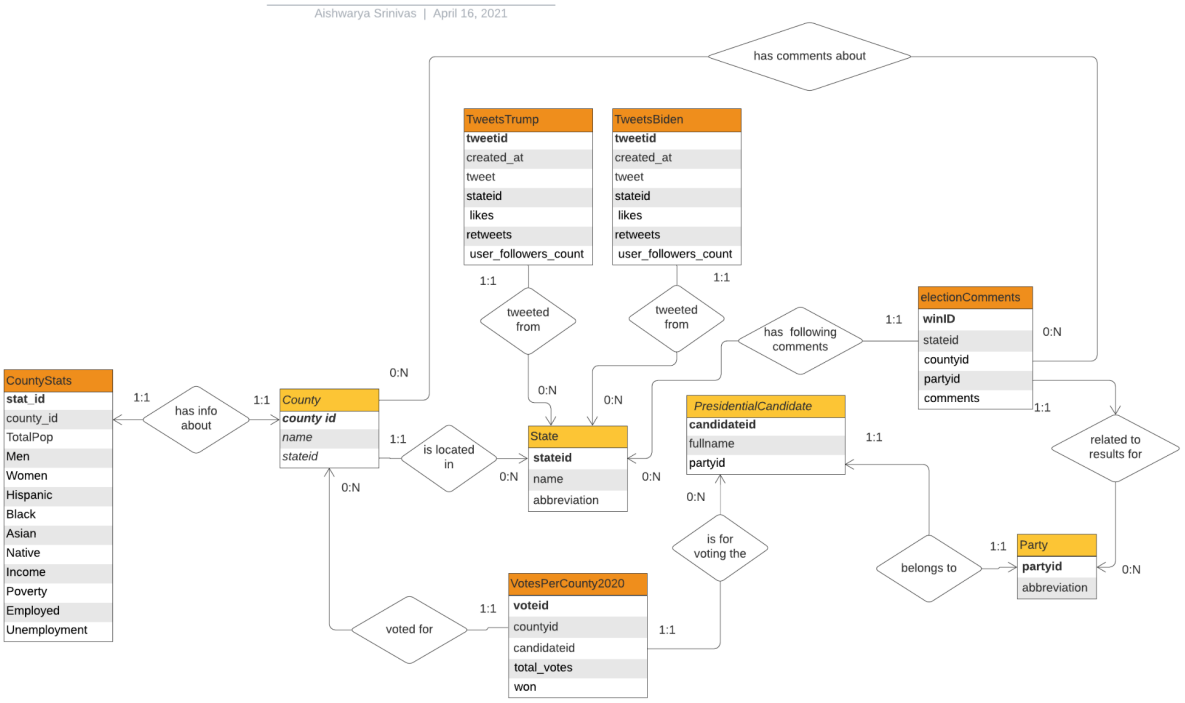
Datasets used

1. President_state.csv
2. President_county.csv
3. President_county_candidate.csv
4. Hashtag_donaldtrump.csv
5. Hashtag_joebiden.csv
6. County_statistics.csv

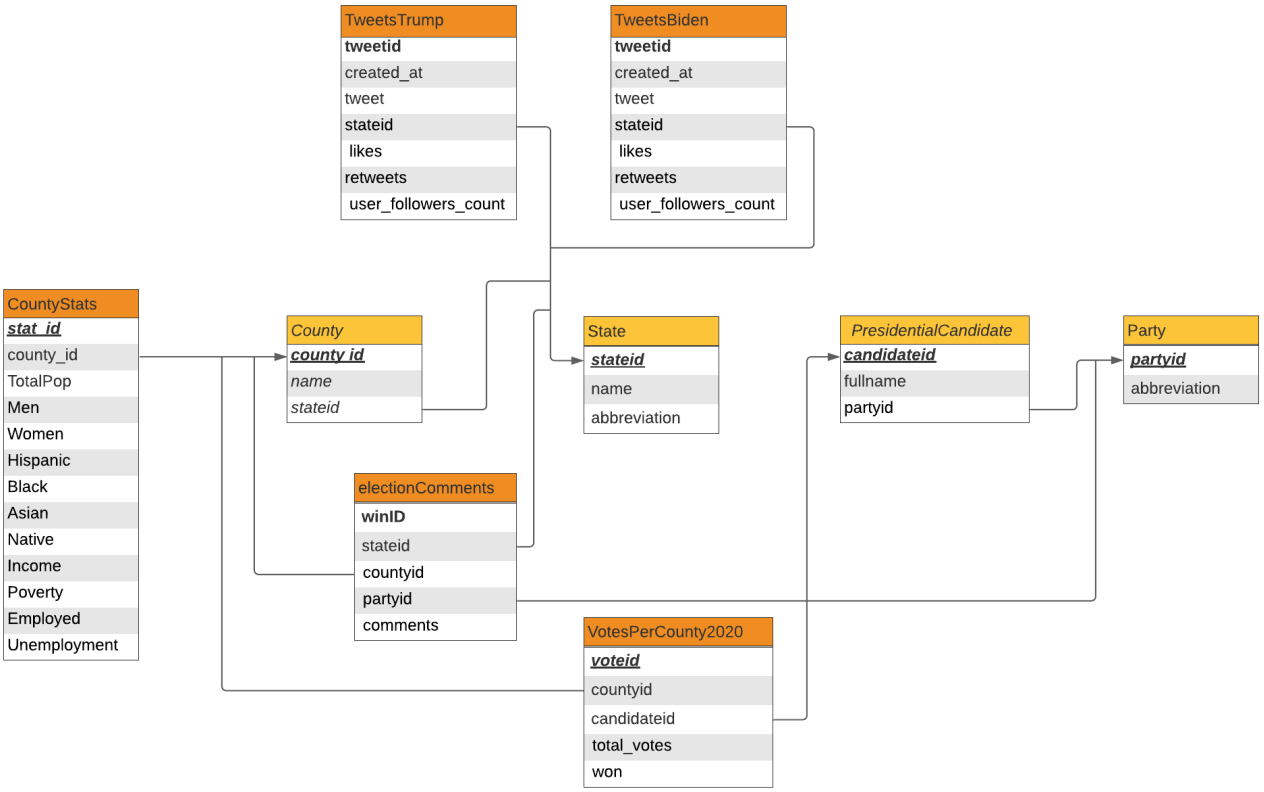
Tables

1. States
2. County
3. Party
4. PresidentialCandidate
5. VotesPerCounty
6. TweetsTrump
7. TweetsBiden
8. CountyStats
9. electionComments

ER Diagram



Relational Diagram



Relation Schema

1. **states**(stateid, name, abbreviation)
2. **county**(countyid, name, stateid)
3. **party**(partyid, abbreviation, name)
4. **presidentialCandidate**(candidateid, fullname, partyid)
5. **votesPerCounty**(voteid, countyid, candidateid,partyid, total_votes,won)
6. **tweetsTrump**(voteid, created_at,tweet,likes, retweets, user_follower_count, stateid)
7. **tweetsBiden**(voteid, created_at,tweet,likes, reteets, user_follower_count, stateid)
8. **countyStats**(stat_id,countyid,TotalPop,Men,Women,Hispanic,White,Black,Native, Asian,Income,Poverty,Employed,Unemployment)
9. **electionComments**(winId, stateid, countyid, partyid, comment)

Normalization and cleanup of raw data

1. We picked necessary data from the datasets that would be useful for this application. So, only a few columns were chosen from each dataset.
2. We noticed that some columns were repeated in the datasets we imported: state names, county names, party names. This now became a duplication and normalization problem, so we dropped these 3 columns where they appeared and

replaced them with stateid, countyid, and partyid. These 3 attributes act as foreign keys referencing the primary keys of the State, County, and Party schema.
3. To account for BCNF in this database, all schemas have a unique primary key, and most dependant non-key attributes have been replaced with foreign keys.
4. The county_statistics.csv had no data for state = Alaska, so we thought it would be best to remove it from the CountyStatisitics table. If the user queries for demographic info about it, the CLI will return the error “Sorry, no results found”.
5. State and Party names were loaded manually through INSERT statements as they are very small tables.

CLI Program Flow

This will outline the main modules in our code base.

Home Page

First, we have a menu that acts as the “homepage” of our CLI that allows users to take four different routes:

1. Get statistics from the 2020 Election
2. Insert Data
3. Observe Data Mining Results
4. Exit

Note: the final trivial route is exiting the program.

The application waits for a numerical input to proceed to the next level of the CLI.

Get Statistics

The types of stats that can be derived from the dataset are presented to the user in the form of a menu with the following options:

1. Voting results for a state: prompts user to enter a state name
2. Voting results for all states
3. Voting results for a county: prompts user to enter a state, and then county
4. Voting results for all counties in a state: prompts user to enter a state
5. Get the most popular tweets from a state: prompts user for state and biden or trump
6. Demographics of a state: prompts user for state
7. Demographics of a county: prompts user for state and county

Insert Data

This allows users to annotate states or counties with comments. The comments are meant to allow users to make notes on the database with the assistance of the statistics. Users can make multiple comments for a state/county.

Data Mining

This will display a table of the most significant factors influencing the election results. Our model predicts who will win in the election, on test data and displays the accuracy of that result.

Libraries

We used libraries for three broad categories: establishing the database connection, computing and using the data model, and testing.

1. Database connection:
    a. ***mysql.connector***: to get access to the connection and cursor objects of the database.
    b. ***getpass***: to prompt the user for username and password rather than require them to type it out manually, which poses a security risk.
    c. Locale: formatting strings
2. Data mining:
    a. **csv**: to convert the results read from the database into a csv
    b. **pandas**: to create the data frame to capture dataset in python
    c. **seaborn**: creating the confusion matrix
    d. **matplotlib**.pyplot: displays charts
    e. Numpy: array operations
    f. **sklearn**: performing the random forest algorithm
3. Testing: **numbers** - to check if numerical inputs are valid

Testing

We approached testing from the following angles:

1. Input validation, and
2. Database connection testing.

Input Validation

First, we examined the types of inputs we would be handling and built test cases for them. Here, we list the cases special to the type of input:

1. Menu options
    a. Input is not a number
    b. Input is a number but does not fall within the required range which is > 1
2. Name of state: input does not exist in the database
3. Name of county: input does not exist in the database
4. Number of tweets:
    a. Input is not a number
    b. Input is a number less than or equal to one
5. Democratic vs Republican (for tweets)
    a. Input for number of tweets is greater than 0

b. The correct option for viewing Trump vs Biden tweets has been chosen

Overlapping cases to modularize:

1. Input is empty. How to handle it:
    a. Display error message
    b. Repeat prompt

Database Connection Testing

Initially, we test to see if the username and password entered are correct. If not we terminate the program. Next, we test to see if the database exists before we proceed. We do this in a try-except block:

```
try:
    cnx = mysql.connector.connect(**config)
except mysql.connector.Error as err:
    if err.errno == errorcode.ER_ACCESS_DENIED_ERROR:
        print("Error: Something is wrong with your user name or password")
        exit()
    elif err.errno == errorcode.ER_BAD_DB_ERROR:
        print("Error: Database does not exist")
        exit()
    else:
        print(err)
else:
```

Data Mining

Question

The question to be addressed: “What factors and demographics influence the win of Biden vs. Trump in the 2020 elections?”

Background

We analyze the demographic data because this is a factor that is different in every county and state. Historically, factors such as race, gender, income, and employment have influenced voter turnout + results, as people will mainly vote for the party that aligns with their policies.

Limitation in the data we chose: However, demographics wholeheartedly will not decide the outcome of an election. Numerous other factors are missing in our datasets, such as how people agree on certain policies, how people voted in previous elections, their attitude

towards the presidential candidate, and many more circumstances. Thus, it is predicted we will not get great accuracy from our model.

Setup of data

The data contains the demographic info for all counties (independent variables) plus the winning party (dependent variable).

Demographic Info = ["population", "men", "women", "white", "black", "hispanic", "asian", "native", "average_income", "poverty", "employed", "unemployed"]

Winning Party labels = ["Democratic", "Republican"]

Main Query to fetch required data

```
select
        ROUND(sum(pop_per_county),2) as 'population',
        ROUND((sum(demographicMen)/sum(pop_per_county))*100,2) as 'men',
        ROUND((sum(demographicWomen)/sum(pop_per_county))*100,2) as 'women',
        ROUND((sum(demographicWhite)/sum(pop_per_county))*100,2) as 'white',
        ROUND((sum(demographicBlack)/sum(pop_per_county))*100,2) as 'black',
        ROUND((sum(demographicHispanic)/sum(pop_per_county))*100,2) as 'hispanic',
        ROUND((sum(demographicAsian)/sum(pop_per_county))*100,2) as 'asian',
        ROUND((sum(demographicNative)/sum(pop_per_county))*100,2) as 'native',
        ROUND(sum(income),2) as 'average_income',
        ROUND((sum(demographicPoverty)/sum(pop_per_county))*100,2) as 'poverty',
        ROUND((sum(demographicEmployed)/sum(pop_per_county))*100,2) as 'employed',
        ROUND((sum(demographicUnemployment)/sum(pop_per_county))*100,2) as
            'unemployed',
        p.partyid as 'winning_party'
        from (
                SELECT  c.name as county,
                        c.countyid as countyid,
                        TotalPop as pop_per_county,
                        (cs.Men) as demographicMen,
                        (cs.Women) as demographicWomen,
                        (cs.White*TotalPop/100) as demographicWhite,
                        (cs.Black*TotalPop/100) as demographicBlack,
                        (cs.Hispanic*TotalPop/100) as demographicHispanic,
                        (cs.Asian*TotalPop/100) as demographicAsian,
                        (cs.Native*TotalPop/100) as demographicNative,
                        Income as income,
                        (cs.Poverty*TotalPop/100) as demographicPoverty,
                        Employed as demographicEmployed,
                        (cs.Unemployment*TotalPop/100) as demographicUnemployment
                FROM CountyStats cs
                inner join County c on (cs.countyid = c.countyid)
        ) as Stats

        inner join County co on (co.countyid = Stats.countyid)
```

```
inner join States ss on (ss.stateid = co.stateid)
inner join VotesPerCounty vpc on (vpc.countyid = Stats.countyid)
inner join Party p on (vpc.partyid = p.partyid)

group by ss.stateid, co.countyid, vpc.won, p.partyid
having vpc.won  = 'True' and p.partyid = 1 or p.partyid = 2
order by ss.stateid asc
```

Note

1. The execution time of this query is 300 ms.
2. All parameters are in float, except for total_population, income and winning_party.

Breakdown

1. Finding percentage for race and gender attributes by state:
    a. **Subquery**:
        i. Select demographic data from the CountyStats table and format percentages so they reflect the actual number of people.
        ii. Also select the county names by doing an inner join with the County table on countyID.
    b. **Main query**:
        i. Summing across all counties and dividing by the sum of each of their total populations and bringing that to a percentage for race, gender, poverty, and employment.
        ii. Summing the total populations in each of the counties.
        iii. Selecting the label attribute from the Party table.
    c. As the query requires information about counties, parties, and votes per county on top of the already included demographic data in the subquery, we do an inner join with the following tables:
        i. County (on countyID),
        ii. States (on stateID),
        iii. VotesPerCounty (on countyID), and
        iv. Party (on partyID).
2. After splitting into training and testing data sets, fill in average values for features holding NULL values (or missing values).

Then, we converted the results into a csv that will be formatted later into a DataFrame.

Technique used to model data

We have multiple continuous variables that we map to 2 discrete values (Democratic, Republic). This is a classification problem so we realised it is best to use random forest classification for this problem.

Random forests is a supervised learning algorithm that uses both regression and classification. Multiple decision trees are created that use the divide and conquer approach to go through each set of entries in the data, and picks the most important features that influenced the result by a large margin [1]. We chose n_estimators = 200, which is the

number of decision trees to train our model. We chose this algorithm because it is flexible and works very well for n inputs and n outputs. This algorithm was relatively fast as well.

Description of Features

1. Population
2. Gender: as a percentage of the total population
    a. Men
    b. Women
3. Race: as a percentage of the total population
    a. White
    b. Black
    c. Hispanic
    d. Asian
    e. Native
4. Average Income for the county
5. Poverty: percentage of total population that live under the poverty line
6. Status of employment:
    a. Employed: as a total number of people
    b. Unemployed: as a percentage of the total population

The label or target used for this algorithm is an attribute we created called “winning_party” that shows the party ID that got the most votes in the county.

Steps

1. Run the query from above to fetch the required data.
2. Generate a CSV from the results called demographics_and_votes.csv.
3. Load the data into a DataFrame, where each index of the array is for each parameter and create a data.
4. Fill any NULL values with the mean value of its respective column.
5. Creates the training and testing sets. We use 70% of the data for training and 30% for testing.
6. Run the Random Forest Classifier on this data.
7. Fit training data to model.
8. Display accuracy and most important features in a tabular format.

Validity of Model

This model yields an accuracy of around 70%.

Results

```
Most important factors deciding the winning party:
+----------------+----------------------+
|  Demographic   | Percentage Influence |
+----------------+----------------------+
|     asian      |       17.708%        |
|     white      |       14.739%        |
|   population   |       10.201%        |
|     black      |        9.586%        |
| average_income |        8.889%        |
|    employed    |        7.309%        |
|    hispanic    |        6.553%        |
|    poverty     |        5.567%        |
|   unemployed   |        5.392%        |
|     women      |        5.351%        |
|      men       |        5.083%        |
|     native     |        3.623%        |
+----------------+----------------------+
Predicting the voting results on test data, we get an accuracy of: 72.773%
```

The percentages show the accuracy score or how much they influenced the results, which is expected. It was expected that % of white, black, and total population would influence the results, but we did not expect that % of the population that was Asian was at the top, or that % of men and women were the least influential.

Reasons for accuracy

1. Our dataset only has around 4000 rows because we only have around 4000 distinct counties.
2. We realise that having too many features could lower the accuracy under this algorithm.

How to improve

1. Improve the correlation between the features and label. As mentioned previously, there many more factors that can affect election results, so using them in our model can improve accuracy.
2. It is best to stick to 5-6 features for this classification technique.
3. Increase training data.
4. Decrease the amount of NULL data we have in our dataset.
5. Use multiple algorithms.

References

[1] *Data to Fish*, 27-Mar-2020. [Online]. Available: https://datatofish.com/random-forest-python/?fbclid=IwAR0Gg6JFKBYm-C7XUnr96uDm5ZZdVQDzCD1oHMUkk4FDngVIIQd6asWsjww. [Accessed: 17-Apr-2021].