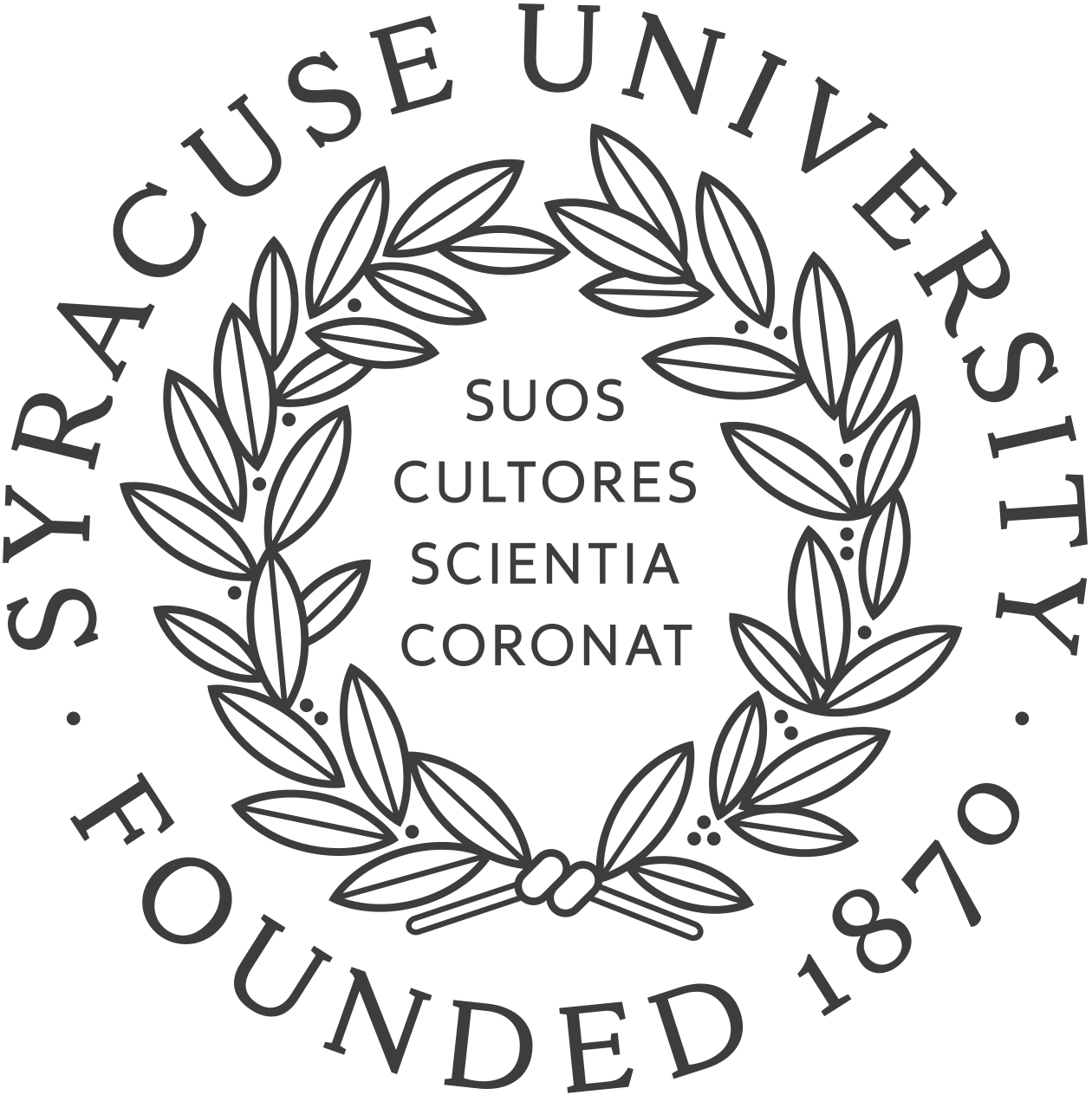
Graphical user interface, text, application

Description automatically generated 

Portfolio Milestone

Daniel Caley

SUID: 352784643

Github: [Masters Applied Data Science Portfolio](https://github.com/dcaley5005/Data_Science/tree/main/Syracuse/Applied%20Data%20Science%20Portfolio)

Table of Contents

[My Data Science Story 4](#_Toc95589381)

[Self-Perceived Barriers 4](#_Toc95589382)

[Applied Data Science Program 4](#_Toc95589383)

[Project 1 – IST 659 Data Administration Concepts & Database Management 6](#_Toc95589384)

[Course Description 6](#_Toc95589385)

[Learning Objectives: 6](#_Toc95589386)

[Deliverables(s) – SQL Database 6](#_Toc95589387)

[Project Process & Development 7](#_Toc95589388)

[*Project Summary:* 7](#_Toc95589389)

[Stakeholders: 7](#_Toc95589390)

[High Level Rules: 7](#_Toc95589391)

[Details of data needed and maintenance: 9](#_Toc95589392)

[Glossary 9](#_Toc95589393)

[Data Questions 11](#_Toc95589394)

[Conceptual Model 14](#_Toc95589395)

[Logical Model 15](#_Toc95589396)

[Physical Database Design 16](#_Toc95589397)

[Utilizing Access & Creating Procedures: 19](#_Toc95589398)

[Creating Views and Functions: 20](#_Toc95589399)

[Answering Data Questions: 30](#_Toc95589400)

[Data Visualizations: 44](#_Toc95589401)

[Reflection: 52](#_Toc95589402)

[Summary: 53](#_Toc95589403)

[Project 2 – IST 718 Big Data Analytics 55](#_Toc95589404)

[Course Description 55](#_Toc95589405)

[Learning Objectives: 55](#_Toc95589406)

[Deliverables(s) – Predicting Zillow House Prices 55](#_Toc95589407)

[Project Process & Development 55](#_Toc95589408)

[Obtain 55](#_Toc95589409)

[Zillow Data 56](#_Toc95589410)

[Census Data 56](#_Toc95589411)

[Scrub 58](#_Toc95589412)

[Zillow Data 58](#_Toc95589413)

[Census Data 60](#_Toc95589414)

[Merging Zillow and Census Data 61](#_Toc95589415)

[Explore 63](#_Toc95589416)

[Arkansas Metro Time Series Plot 63](#_Toc95589417)

[Arkansas Metro Percentage Return (1997 – 2019) 64](#_Toc95589418)

[Arkansas Metro Results (2010 – 2019) 65](#_Toc95589419)

[Arkansas Metro Population and Household Median Income (2010 – 2019) 65](#_Toc95589420)

[Bonus Geographic Visualization –Median Housing Price 66](#_Toc95589421)

[Bonus Geographic Visualization – Population 67](#_Toc95589422)

[Bonus Geographic Visualization - Household Median Income 68](#_Toc95589423)

[Modeling 69](#_Toc95589424)

[Historic Risk and Return - USA 69](#_Toc95589425)

[Historic Risk and Return - Arkansas 70](#_Toc95589426)

[Forecasting Arkansas Return - Scrubing 71](#_Toc95589427)

[Forecasting Arkansas Return - Results 74](#_Toc95589428)

[Project 3 – IST 664 Natural Language Processing 78](#_Toc95589429)

[Course Description 78](#_Toc95589430)

[Learning Objectives: 78](#_Toc95589431)

[Deliverables(s) – Classifying IMDB Movie Reviews 78](#_Toc95589432)

[Project Process & Development 79](#_Toc95589433)

[Overall 79](#_Toc95589434)

[Reviewing the Data 80](#_Toc95589435)

[Tokenized data 82](#_Toc95589436)

[Filtered and tokenized data 84](#_Toc95589437)

[Pre-processed and tokenized data 84](#_Toc95589438)

[Feature Engineering 88](#_Toc95589439)

[Bag of Words & Unigrams Features 88](#_Toc95589440)

[Bigrams Features 88](#_Toc95589441)

[Part-of-speech Features 89](#_Toc95589442)

[Sentiment Lexicon (LIWC) 90](#_Toc95589443)

[Subjectivity 90](#_Toc95589444)

[Experiments 91](#_Toc95589445)

[Bayes Classifier 91](#_Toc95589446)

[Cross Validation 91](#_Toc95589447)

[Sci-kit Learn - Advance Task 91](#_Toc95589448)

[Results 92](#_Toc95589449)

[Final Thoughts 95](#_Toc95589450)

# My Data Science Story

When Michael Burry, the hedge fund manager who combed through mountains and mountains of mortgage data and foresaw the worst economic crisis since the great depression, I knew then that I wanted to be a Data Scientist. At the time though I didn’t know what the field was called, let alone if dedicated roles like that existed. So, I pursued a trading position because that what Michael was doing. I went to work for a company at ground zero of the financial crises, I decided I wanted to run towards the burning building, instead of away. I learned SQL, Advance Excel Techniques, and VBA to scrub, analyze, and bring value to the mortgage market. As I grew in the role, I realized that the Fantasy of becoming a trader did not meet up to the reality. I swallowed a tough pill and looked for companies more into Data. There at my lowest moment in my Career I found a Data Analytics role and my journey truly begun in the field.

I absolutely love every minute of obtaining, scrubbing, transforming, and visualizing data. I felt like the fantasy finally met up to the reality, but a sweeping fear came across me. I didn’t know Machine Learning, I didn’t understand Bayesian Hypothesis testing, I didn’t appreciate base statistics as I should have. The Inferential and Predictive Statistics was lacking, and I feared that I would get stuck and forgotten in the world that I loved so much. There and then I understood that I had to lean into my passion, and I applied for a master’s in applied data science at Syracuse University.

# Self-Perceived Barriers

Coding didn’t seem like something my future would hold. I looked to Software Engineers, not analysts, to discourage my ability to want to learn coding. Through my career and Syracuse, I was able to demystify this barrier and really unlock a lot of the powers of these tools. This led to a snowball effect where Libraries and Packages became these easy things to load and apply on Data. This too led to using advance statistics, Bayesian theory, Linear Regression, P-Values, and all of these terms that felt so scarry I now had confidence.

# Applied Data Science Program

The Applied Data Science program at Syracuse University's School of Information Studies equips students with the opportunity to develop analytic, technical, and managerial skills in order to contribute measurable impacts in a highly competitive job market. The focus of the program is encapsulated in the word "Applied": students are taught to collect, manage, analyze and develop insights that can be communicated clearly to a broad audience of interested parties. Through courses such as Data Administration Concepts & Database Management (IST 659), Data Analytics (IST 707), Scripting for Data Analysis (IST 652) and Advanced Database Management (IST 769), I learned basic data structure (from byte to bit); how relational databases are built using SQL, Excel and Microsoft Access, meeting the business needs of organizations; how data mining techniques can be used to solve classification problems using R Studio; and, scripting in Python to explore a dataset to gain insights; using visualization to explore data and to communicate findings to a broad spectrum of audiences.

The Applied Data Science Program has seven learning objectives which were exemplified by the applications in this portfolio:

1. Describe a broad overview of the major practice areas in data science.
2. Collect and organize data.
3. Identify patterns in data via visualization, statistical analysis, and data mining.
4. Develop alternative strategies based on the data.
5. Develop a plan of action to implement the business decisions derived from the analyses.
6. Demonstrate communication skills regarding data and its analysis for relevant professionals in their organization.
7. Synthesize the ethical dimensions of data science practice.

# Project 1 – IST 659 Data Administration Concepts & Database Management

## Course Description

IST 659 is an introductory course to database management systems. This course examines data structures, file organizations, concepts, and principles of database management systems (DBMS) as well as data analysis, database design, data modeling, database management, and database implementation. More specifically, it introduces hierarchical, network, and relational data models; entity-relationship modeling; basics of Structured Query Language (SQL); data normalization; and database design. Using Microsoft’s Access and SQL Server DBMSs as implementation vehicles, this course provides hands-on experience in database design and implementation through assignments, lab exercises, and course projects. This course also introduces advanced database concepts such as transaction management and concurrency control, distributed databases, multitier client/server architectures, web-based database applications, data warehousing, and NoSQL.

## Learning Objectives:

After taking this course, the students will be able to:

* Describe fundamental data and database concepts
* Explain and use the database development lifecycle
* Create databases and database objects using popular database management system products
* Solve problems by constructing database queries using Structured Query Language (SQL)
* Design databases using data modeling and data normalization techniques
* Develop insights into future data management tool and technique trends
* Recommend and justify strategies for managing data security, privacy, audit/control, fraud detection, backup and recovery
* Critique the effectiveness of DBMS in computer information systems

## Deliverables(s) – SQL Database

Created a research focus database that looked at real data from the census bureau and CDC covid data. The goal of the project to dive into the socioeconomic impact of COVID on the American society.

I created a fictional non-profit that needed this data in a structured way in order to further this research. First a conceptual model of the database was created followed by a logical model with the objective of understanding how the data all ties together.

SQL Server Management Studio was used to create the tables and populate the data, all using SQL. Personalized views were constructed for the non-profit to easily access whats important to them, like average covid rates by state or ethnicity.

## Project Process & Development

### Project Summary:

A non for profit wants to understand the impact of COVID related cases and deaths across America. Currently they have daily csv files by county of the number of COVID cases and deaths starting from May 2020 until present. They are finding that they need all this data in one place and simply aggregating the daily csv files into a single excel file is not scalable to perform analysis. In addition to their COVID data, they want to bring in additional Census Bureau data to help make their analysis even richer. The data includes population, household median income, and race by ZCTA. They will also need the ability to map ZCTA to Counties, Cities, and State. The final outcome for this project will allow workers of the non-profit to have all of their data in one place in order to perform analysis.

### Stakeholders:

The stakeholders in this project are the non-profit who is interested in the impact of COVID on race and socioeconomic status in America. Inside of the non-profit the analysts working there will use the data to create presentations and research material surrounding the effects of the Coronavirus on different segments of the US population. Ultimately to help inform businesses, local municipalities, and news organizations that are in need of additional funding and to help combat the coronavirus impact on America.

### High Level Rules:

* COVID cases and deaths are broken out daily by County. They only report daily, not twice a day.
* Cases and deaths might not be reported if there was no cases and death on that day for that county. The CSV by the non- profit will fill in the missing days with 0 cases and deaths by county.
* Census Bureau data is created at the A. This is different from postal Zip Code. There can be many Zip Codes underlying a ZCTA. We are not working with Zip Code in this dataset so we will not have to worry. Just need to know that these are 2 different ways to break out County level data.
* A Geo County State map will be required to connect the County to ZCTA.
  + The Geo County State map will include a county id and State.
  + The hierarchy for the multiple geographical attributes in this table are the following, from lowest level to the top:
    1. ZCTA
    2. City
    3. County
    4. State
  + Said differently there can be many ZCTA underlying a City, there can be many Cities underlying a County, and there can be many Counties underlying a State.
* The Census Bureau provides population by age group and by ZCTA.
  + Each ZCTA will have multiple age groups and consist of 6 groups.
* The Census Bureau also provides Household Median Income by ZCTA.
* The Census Bureau also has race information by ZCTA. The race information provided in a race category. The Race categories consist of White, Asian, Black or African American, and Other.

### Details of data needed and maintenance:

The following data will be needed:

1. COVID data will be needed and is tracked by the non-profit. They will be responsible for creating the CSV files. They will also be responsible for making sure that all counties exist in the CSV file with a 0 value for cases and deaths in the event there are none. The data engineer, myself, will be responsible for uploading the data into the database on a daily basis.
2. The Census Bureau data is ingested from an API. This data is updated yearly at the end of Quarter 1 or the beginning of Quarter 2. It’s the responsibility of the Data Engineer to swap out the previous year’s data with the current.
3. The non for profit does not want stale data from the Census Bureau and only wants the new data.

### Glossary

A **ZCTA** stands for Zip Code Tabulation Areas and are generalized areal representations of United States Postal Service Zip Code service areas. ZCTA are trademarked by the U.S. Census Bureau. The hierarchy is that one or many Zip Codes can underly a ZCTA.

A **City** an inhabited place of greater size, population or importance. In the hierarchy many zip codes can exist inside of a city.

A **County** is political and administrative division of a state, providing certain local governmental services. There are many cities to a county.

A **State** is a territory considered as an organized political community under one government. There are many counties to a single State.

**Household Median Income** is the income level earned by a given household. The formal definition is the income cut-off where half of the households earn ore, and half earn less. In this case Household Median Income is a widely used metric when defining socioeconomic areas in business and non for profits.

**Population** is all inhabitants of a particular Zip Code, ZCTA, city, county, or state.

**Race** is a grouping of people who identify with each other on the basis of shared attributes that distinguish them from other groups such as a common set of traditions, ancestry, language, history, society or skin color. For the census data Race consists of 6 groups. They are the following:

1. White alone
2. Black or African American alone
3. American Indian and Alaska Native alone
4. Asian alone
5. Native Hawaiian and Other Pacific Islander alone
6. Other Race alone

**Age Group** consist of 23 groups. They are the following:

1. Under 5 years
2. 5 to 9 years
3. 10 to 14 years
4. 15 to 17 years
5. 18 and 19 years
6. 20 years
7. 21 years
8. 22 to 24 years
9. 25 to 29 years
10. 30 to 34 years
11. 35 to 39 years
12. 40 to 44 years
13. 45 to 49 years
14. 50 to 54 years
15. 55 to 59 years
16. 60 and 61 years
17. 62 to 64 years
18. 65 and 66 years
19. 67 to 69 years
20. 70 to 74 years
21. 75 to 79 years
22. 80 to 84 years
23. 85 years and over

**Cases** refer to the number of COVID related cases.

**Deaths** refer to the number of COVID related deaths.

**COVID** a highly contagious respiratory disease caused by the SARS-CoV-2 virus. Also known for causing a pandemic across the world that has disrupt economies, impacted a way of life, and caused many deaths.

### Data Questions

1. What is the trailing 7-day COVID cases and deaths from May 2020 until current?
   * Calculate the trailing 7-day cases divided by population.
   * Calculate the trailing 7-day deaths divided by population.
2. What is the Population and Household Median Income for the ZCTA 85203, Mesa AZ.
   * This should be by ZCTA, County, & State
   * What Percentage Lives in 85203, Maricopa vs the state?
3. What is the total population By County, Include State as a column?
   * What is the top 25 populated counties?
   * What is the trailing 7 days COVID cases divided by population for the top 25 counties?
4. What is the weighted average household median income by County?
   * What’s the top 25 highest weighted average household median income by population?
   * What is the total population of these counties?
   * What’s the bottom 25 weighted average household median income by population?
   * What is the trailing 7-day COVID cases divide by population?
   * What does the top 25 highest weighted average household median income look compare to the bottom 25?
5. For ZCTA 85203, Arizona, what is the population broken out by age group?
   * How does that compare to the County?
   * How does that compare to the State?
6. What where the top 25 Counties who had the highest deaths by percentage of population?
   * The population must be over 1000 people.
   * Include their household median income.
   * Not looking at New York
7. Group the age groups that are considered the high-risk age categories for COVID. That is age groups over 65 and under 5.
   * Make this a calculated field that high risk or low risk.
   * Total the population by high risk and low risk by county.
   * Any counties with a population that is greater than 30% of the total population what is the number of COVID cases and deaths by total population vs low risk counties.
   * Group these counties into 2 rows, high risk and low risk. How do they total cases and deaths compare to each other?
8. How many Deaths Vs. Total Population
   * Compare 2 rows below 30% and above 30% and how do the total cases and deaths by population compare to each other?

### Conceptual Model

Diagram

Description automatically generated

### Logical Model

Diagram

Description automatically generated

### Physical Database Design

-- Creating Tables, Views, procedure, and functions.

-- Creating Drops

--Drop Tables

Drop table if exists dc\_census\_bureau\_population

go

drop table if exists dc\_census\_bureau\_race

go

Drop table if exists dc\_covid

go

Drop table if exists dc\_census\_bureau\_income

go

Drop table if exists dc\_geo\_county\_state\_map

go

-- Creating database tables

-- Creating daily covid & death table

CREATE TABLE dc\_geo\_county\_state\_map(

-- Creating Columns

county\_id varchar(5),

county varchar(255) NOT NULL,

state varchar(2) NOT NULL,

-- Applying Constraints

CONSTRAINT PK\_dc\_geo\_county\_state\_map PRIMARY KEY (county\_id)

)

go

CREATE TABLE dc\_covid (

-- Creating Columns

covid\_id varchar(10),

county\_id varchar(5) NOT NULL,

date date NOT NULL,

cases int NOT NULL,

deaths int NOT NULL,

-- Applying Constraints

CONSTRAINT PK\_dc\_covid PRIMARY KEY (covid\_id),

CONSTRAINT FK1\_dc\_covid FOREIGN KEY (county\_id) REFERENCES

dc\_geo\_county\_state\_map(county\_id)

)

go

CREATE TABLE dc\_census\_bureau\_income (

-- Creating Columns

zcta varchar(5),

county\_id varchar(5) NOT NULL,

household\_median\_income numeric(10,2),

-- Applying Constraints

CONSTRAINT PK\_dc\_census\_bureau\_income PRIMARY KEY (zcta),

CONSTRAINT FK\_dc\_census\_bureau\_income FOREIGN KEY (county\_id) REFERENCES

dc\_geo\_county\_state\_map(county\_id)

)

go

CREATE TABLE dc\_census\_bureau\_race (

-- Creating Columns

race\_id varchar(8),

zcta varchar(5),

race varchar(255),

population int,

-- Applying Constraints

CONSTRAINT PK\_dc\_census\_bureau\_race PRIMARY KEY (race\_id),

CONSTRAINT FK\_dc\_census\_bureau\_race FOREIGN KEY (zcta) REFERENCES

dc\_census\_bureau\_income(zcta)

)

go

CREATE TABLE dc\_census\_bureau\_population (

-- Creating Columns

population\_id varchar(7),

zcta varchar(5),

age\_group varchar(255),

population int

-- Applying Constraints

CONSTRAINT PK\_dc\_bureau\_population PRIMARY KEY (population\_id),

CONSTRAINT FK\_dc\_bureau\_population FOREIGN KEY (zcta) REFERENCES

dc\_census\_bureau\_income(zcta)

)

go

--- Inserting records into tables. Total recoreds is over 1 million.

--- Therefore I will load the records through a bulk upload process.

--- The below will only show 5 records for each table.

--- Inserting Records for geo\_county\_state\_map

INSERT INTO dc\_geo\_county\_state\_map(county\_id, county, state)

VALUES ('01001', 'Autauga County', 'AL'),

('01003', 'Baldwin County', 'AL'),

('01005', 'Barbour County', 'AL'),

('01007', 'Bibb County', 'AL'),

('01009', 'Blount County', 'AL')

go

-- Look to see if the insert worked

select \* from dc\_geo\_county\_state\_map

Graphical user interface, table

Description automatically generated

--- Inserting Records for census\_bureau\_income

INSERT INTO dc\_census\_bureau\_income(zcta, county\_id, household\_median\_income)

VALUES ('36003', '01001', '37000'),

('36480', '01003', '27461'),

('36005', '01005', '49722'),

('35034', '01007', '39087'),

('35013', '01009', '0')

go

-- Look to see if the insert worked

select \* from dc\_census\_bureau\_income

Table

Description automatically generated

--- Inserting Records for covid

INSERT INTO dc\_covid(covid\_id, county\_id, date, cases, deaths)

VALUES ('0000009481', '01001', '2020-03-24', '1', '0'),

('0000001935', '01003', '2020-03-14', '1', '0'),

('0000028399', '01005', '2020-04-03', '1', '0'),

('0000019658', '01007', '2020-03-30', '2', '0'),

('0000010838', '01009', '2020-03-25', '1', '0')

go

-- Look to see if the insert worked

select \* from dc\_covid

A picture containing text, cabinet

Description automatically generated

--- Inserting Records for population

INSERT INTO dc\_census\_bureau\_population(population\_id, zcta, age\_group, population)

VALUES ('360031', '36003', 'Under 5 years', '111'),

('364801', '36480', 'Under 5 years', '28'),

('360051', '36005', 'Under 5 years', '57'),

('350341', '35034', 'Under 5 years', '547'),

('350131', '35013', 'Under 5 years', '0')

go

-- Look to see if the insert worked

select \* from dc\_census\_bureau\_population

Table

Description automatically generated

--- Inserting Records for race

INSERT INTO dc\_census\_bureau\_race(race\_id, zcta, race, population)

VALUES ('36003101', '36003', 'Black or African American alone', '1102'),

('36480100', '36480', 'White alone', '1389'),

('36005102', '36005', 'American Indian and Alaska Native alone', '0'),

('35034100', '35034', 'White alone', '2942'),

('35013105', '35013', 'Other Race alone', '0')

go

-- Look to see if the insert worked

select \* from dc\_census\_bureau\_race

Graphical user interface, text, application

Description automatically generated

### Utilizing Access & Creating Procedures:

--- Form made from Access to help fix issues in the database

Graphical user interface

Description automatically generated

--- Creating a procedure to update a COVID cases based from the county id and date.

--- The first parameter is the user name for the user to change

--- The second is the new email address

CREATE PROCEDURE dc\_ChangeCovidCases(@covid\_cases int,@county\_id varchar(5), @covid\_date date)

AS

BEGIN

UPDATE dc\_covid SET cases = @covid\_cases

WHERE county\_id = @county\_id and date = @covid\_date

END

GO

Graphical user interface, application

Description automatically generated

### Creating Views and Functions:

-- Before answering the questions going to create a series of views and functions to help answer the questions.

--- Finding total covid cases and deaths by county.

drop view if exists dc\_covid\_county

go

CREATE VIEW dc\_covid\_county as

select dc\_covid.county\_id,

dc\_geo\_county\_state\_map.county,

dc\_geo\_county\_state\_map.state,

sum(cases) total\_cases,

sum(deaths) total\_deaths

from dc\_covid

join dc\_geo\_county\_state\_map

on dc\_covid.county\_id = dc\_geo\_county\_state\_map.county\_id

group by dc\_covid.county\_id,

dc\_geo\_county\_state\_map.county,

dc\_geo\_county\_state\_map.state

go

--- Checking to see if the view worked

select \* from dc\_covid\_county

A picture containing table

Description automatically generated

--- Finding daily covid cases nation wide from May 2020 until current.

drop view if exists dc\_daily\_covid

go

CREATE VIEW dc\_daily\_covid as

with daily\_covid as (

Select county\_id,

date,

sum(cases) daily\_cases,

sum(deaths) daily\_deaths

from dc\_covid

-- We have to go 7 days before May as we want a rolling 7 days.

-- SQL will begin the sum but the first 6 records won't be a rolling 7 days

where date between '20200424' and GETDATE()

group by county\_id, date

),

trailing\_covid as (

select \*,

sum(daily\_cases) over (partition by county\_id order by date rows between 6 preceding and current row) t7d\_cases,

sum(daily\_deaths) over (partition by county\_id order by date rows between 6 preceding and current row) t7d\_deaths

from daily\_covid

)

select \* from trailing\_covid

where date >= '20200501'

go

--- Checking our view

select \* from dc\_daily\_covid order by county\_id, date

Table

Description automatically generated

-- creating a zcta, county, state view with total population and median income

drop view if exists dc\_geo\_population

go

CREATE VIEW dc\_geo\_population as

Select dc\_census\_bureau\_income.zcta,

dc\_census\_bureau\_income.county\_id,

dc\_geo\_county\_state\_map.county,

dc\_geo\_county\_state\_map.state,

sum(dc\_census\_bureau\_population.population) population,

dc\_census\_bureau\_income.household\_median\_income

from dc\_census\_bureau\_income

join dc\_geo\_county\_state\_map

on dc\_census\_bureau\_income.county\_id = dc\_geo\_county\_state\_map.county\_id

join dc\_census\_bureau\_population

on dc\_census\_bureau\_income.zcta = dc\_census\_bureau\_population.zcta

group by dc\_census\_bureau\_income.zcta,

dc\_census\_bureau\_income.county\_id,

dc\_geo\_county\_state\_map.county,

dc\_geo\_county\_state\_map.state,

dc\_census\_bureau\_income.household\_median\_income

go

-- Checking our view

select \* from dc\_geo\_population

where state = 'NY' order by zcta

Table

Description automatically generated

-- Creating a view for age groups that includes a covid\_age\_type

-- and attributes like county and state

drop view if exists dc\_age\_group

go

create view dc\_age\_group as

select

dc\_census\_bureau\_population.zcta,

case

when age\_group in ('Under 5 years','65 and 66 years',

'67 to 69 years','70 to 74 years',

'75 to 79 years','80 to 84 years',

'85 years and over')

then 'High Risk Age Group'

else 'Low Risk Age Group'

end as covid\_age\_group,

dc\_census\_bureau\_population.age\_group,

dc\_geo\_county\_state\_map.county\_id,

dc\_geo\_county\_state\_map.county,

dc\_geo\_county\_state\_map.state,

dc\_census\_bureau\_population.population,

case

when age\_group = '85 years and over'

then 85

else left(ltrim(right(age\_group,8)),2)\* 1

end

as sort\_key

from dc\_census\_bureau\_population

left join dc\_census\_bureau\_income

on dc\_census\_bureau\_population.zcta = dc\_census\_bureau\_income.zcta

left join dc\_geo\_county\_state\_map

on dc\_census\_bureau\_income.county\_id = dc\_geo\_county\_state\_map.county\_id

go

-- Checking our view

select \* from dc\_age\_group

order by zcta, sort\_key

Table

Description automatically generated

Drop function if exists dbo.dc\_GetPopulation

go

CREATE FUNCTION dbo.dc\_GetPopulation(@countyid varchar(5))

RETURNS INT AS

BEGIN

DECLARE @Popcounty int

SELECT @Popcounty = SUM(population) from dc\_geo\_population where county\_id = @countyid

RETURN @Popcounty

END

Go

--- Checking to see if the function works. Maricopa County is 04013

select top 1 county\_id, dbo.dc\_GetPopulation(county\_id) county\_population

from dc\_covid

where county\_id = '04013'



--- Create a function that, when given the zcta, returns the total population by County

Drop function if exists dbo.dc\_ZCTA\_CountyPop

go

CREATE FUNCTION dbo.dc\_ZCTA\_CountyPop(@zcta varchar(5))

RETURNS INT AS

BEGIN

DECLARE @Countypop int

SELECT @Countypop = SUM(population) from dc\_geo\_population where county\_id = (select county\_id from dc\_geo\_population where zcta = @zcta)

RETURN @Countypop

END

Go

-- Checking to see if the function works. Using ZCTA 85203

select zcta, dbo.dc\_ZCTA\_CountyPop(zcta) county\_population

from dc\_geo\_population

where zcta = '85203'



--- Create a function that, when given the zcta, returns the total population by State

Drop function if exists dbo.dc\_ZCTA\_StatePop

go

CREATE FUNCTION dbo.dc\_ZCTA\_StatePop(@zcta varchar(5))

RETURNS INT AS

BEGIN

DECLARE @Statepop int

SELECT @Statepop = SUM(population) from dc\_geo\_population where state = (select state from dc\_geo\_population where zcta = @zcta)

RETURN @Statepop

END

Go

-- Checking to if the function works

Select zcta, dbo.dc\_ZCTA\_StatePop(zcta) state\_population

from dc\_geo\_population

where zcta = '85203'



--- Create a function that, when given the zcta, returns the total Household Median Income by County

Drop function if exists dbo.dc\_County\_MedianIncome

go

CREATE FUNCTION dbo.dc\_County\_MedianIncome(@zcta varchar(5))

RETURNS INT AS

BEGIN

DECLARE @CountyIncome numeric(10,2)

SELECT @CountyIncome = SUM(household\_median\_income\*population) / sum(population) from dc\_geo\_population where county\_id = (select county\_id from dc\_geo\_population where zcta = @zcta)

RETURN @CountyIncome

END

Go

-- Checking to if the function works

Select zcta, household\_median\_income, dbo.dc\_County\_MedianIncome(zcta) County\_MedianIncome

from dc\_geo\_population

where zcta = '85203'



--- Create a function that, when given the zcta, returns the Household Median Income by State

Drop function if exists dbo.dc\_State\_MedianIncome

go

CREATE FUNCTION dbo.dc\_State\_MedianIncome(@zcta varchar(5))

RETURNS INT AS

BEGIN

DECLARE @StateIncome numeric(10,2)

SELECT @StateIncome = SUM(household\_median\_income\*population) / sum(population) from dc\_geo\_population where state = (select state from dc\_geo\_population where zcta = @zcta)

RETURN @StateIncome

END

Go

-- Checking to if all these median income function works

Select zcta, household\_median\_income, dbo.dc\_State\_MedianIncome(zcta) State\_MedianIncome

from dc\_geo\_population

where zcta = '85203'

Graphical user interface, text, application

Description automatically generated

Drop function if exists dbo.dc\_County\_AgeGroup

go

CREATE FUNCTION dbo.dc\_County\_AgeGroup(@zcta varchar(5), @agegroup varchar(255))

RETURNS INT AS

BEGIN

DECLARE @Countypop int

SELECT @Countypop = sum(population) from dc\_age\_group where county =

(select county

from dc\_age\_group

where zcta = @zcta and age\_group = @agegroup)

and age\_group = @agegroup

RETURN @Countypop

END

Go

-- Checking to if all the age group county function works

Select zcta, population, age\_group, dbo.dc\_County\_AgeGroup(zcta, age\_group) County\_Population

from dc\_age\_group

where zcta = '85203'

order by sort\_key

Table

Description automatically generated

--- Create a function that, when given the zcta and age group, returns the States total population

Drop function if exists dbo.dc\_State\_AgeGroup

go

CREATE FUNCTION dbo.dc\_State\_AgeGroup(@zcta varchar(5), @agegroup varchar(255))

RETURNS INT AS

BEGIN

DECLARE @Statepop int

SELECT @statepop = sum(population) from dc\_age\_group where state =

(select state

from dc\_age\_group

where zcta = @zcta and age\_group = @agegroup)

and age\_group = @agegroup

RETURN @statepop

END

Go

-- Checking to if all the age group state function works

Select zcta, population, age\_group, dbo.dc\_State\_AgeGroup(zcta, age\_group) State\_Population

from dc\_age\_group

where zcta = '85203'

order by sort\_key

Table

Description automatically generated with medium confidence

### Answering Data Questions:

-- 1. What is the daily and trailing 7-day COVID cases and deaths from May 2020 until current?

-- Calculate the trailing 7-day cases divided by population.

-- Calculate the trailing 7-day deaths divided by population.

-- 1. What is the daily and trailing 7-day COVID cases and deaths May 2020 until current?

-- Going to create 2 temporay views, commit the tables to daily cases and deaths view, and then drop the first 2 views

-- Droping views to not muddy up the database. Was only using them as suedo temp table.

-- Calculate the trailing 7-day cases divided by population.

-- Calculate the trailing 7-day deaths divided by population.

-- 2. What is the Population and Household Median Income for the ZCTA 85203.

-- This should be by ZCTA, County, & State

-- What Percentage Lives in 85203, Maricopa vs the state

Select zcta,

county,

state,

population ZCTA\_Population,

dbo.dc\_ZCTA\_CountyPop(zcta) County\_Population,

dbo.dc\_ZCTA\_StatePop(zcta) State\_Population,

household\_median\_income ZCTA\_MedianIncome,

dbo.dc\_County\_MedianIncome(zcta) County\_MedianIncome,

dbo.dc\_State\_MedianIncome(zcta) State\_MedianIncome

from dc\_geo\_population

where zcta = '85203'

Select zcta,

county,

state,

population \* 1.0 / dbo.dc\_ZCTA\_StatePop(zcta) ZCTA\_perc\_vs\_state,

dbo.dc\_ZCTA\_CountyPop(zcta) \* 1.0 / dbo.dc\_ZCTA\_StatePop(zcta) County\_perc\_vs\_state

from dc\_geo\_population

where zcta = '85203'

Graphical user interface, text, application

Description automatically generated

--- 3. What is the total population By County?

--- What is the top 25 populated counties?

--- What’s their Weighted Average Household Median Income by population?

--- What is the trailing 7 days COVID cases divided by population?

select state,

county,

SUM(population) total\_population

from dc\_geo\_population

group by state, county

select top 25 state,

county,

SUM(population) total\_population

from dc\_geo\_population

group by state, county

order by total\_population desc

select top 25 state,

county,

SUM(population) total\_population

from dc\_geo\_population

group by state, county

order by total\_population desc

go

with top\_25\_county as (

select top 25 state,

county,

county\_id,

SUM(population) total\_population

from dc\_geo\_population

group by state, county, county\_id

order by total\_population desc)

select date,

state,

county,

top\_25\_county.county\_id,

daily\_cases,

t7d\_cases,

t7d\_deaths,

t7d\_deaths \* 1.0 / t7d\_cases deaths\_cases\_perc,

t7d\_cases \* 1.0 / total\_population cases\_by\_population

from top\_25\_county

left join dc\_daily\_covid

on top\_25\_county.county\_id = dc\_daily\_covid.county\_id

order by total\_population desc, date

Table

Description automatically generated with medium confidence

Table

Description automatically generated

Table

Description automatically generated

--4. What is the weighted average household median income by County?

-- What’s the top 25 highest weighted average household median income by population?

-- What is the total population of these counties?

-- What’s the bottom 25 weighted average household median income by population?

-- What is the trailing 7-day COVID cases divide by population?

-- What does the top 25 highest weighted average household median income look compare to the bottom 25?

select top 25 state,

county,

SUM(population) total\_population,

coalesce(sum(population \* household\_median\_income) / nullif(sum(population),0),0) weighted\_avg\_income

from dc\_geo\_population

group by state, county

order by weighted\_avg\_income desc

select top 25 state,

county,

SUM(population) total\_population,

coalesce(sum(population \* household\_median\_income) / nullif(sum(population),0),0) weighted\_avg\_income

from dc\_geo\_population

where state != 'PR'

group by state, county

having coalesce(sum(population \* household\_median\_income) / nullif(sum(population),0),0) > 0

order by weighted\_avg\_income

go

with top\_25\_county as (

select top 25 state,

county,

county\_id,

SUM(population) total\_population,

coalesce(sum(population \* household\_median\_income) / nullif(sum(population),0),0) weighted\_avg\_income

from dc\_geo\_population

where state != 'PR'

group by state, county, county\_id

order by weighted\_avg\_income desc),

top\_25\_cases as (

select date,

state,

county,

top\_25\_county.county\_id,

total\_population,

weighted\_avg\_income,

daily\_cases,

t7d\_cases,

t7d\_cases \* 1.0 / total\_population cases\_by\_population

from top\_25\_county

left join dc\_daily\_covid

on top\_25\_county.county\_id = dc\_daily\_covid.county\_id

),

top\_25\_totaled as (

select 'Top 25 Counties' as covid\_impact,

date,

sum(total\_population) total\_population,

sum(total\_population \* weighted\_avg\_income) / sum(total\_population) weighted\_avg\_income,

sum(t7d\_cases) t7d\_cases,

sum(t7d\_cases) \* 1.0 / sum(total\_population) cases\_by\_population

from top\_25\_cases

group by date

),

bottom\_25\_county as (

select top 25 state,

county,

county\_id,

SUM(population) total\_population,

coalesce(sum(population \* household\_median\_income) / nullif(sum(population),0),0) weighted\_avg\_income

from dc\_geo\_population

where state != 'PR'

group by state, county, county\_id

order by weighted\_avg\_income),

bottom\_25\_cases as (

select date,

state,

county,

bottom\_25\_county.county\_id,

total\_population,

weighted\_avg\_income,

daily\_cases,

t7d\_cases,

t7d\_cases \* 1.0 / total\_population cases\_by\_population

from bottom\_25\_county

left join dc\_daily\_covid

on bottom\_25\_county.county\_id = dc\_daily\_covid.county\_id

),

bottom\_25\_totaled as (

select 'Bottom 25 Counties' as covid\_impact,

date,

sum(total\_population) total\_population,

sum(total\_population \* weighted\_avg\_income) / sum(total\_population) weighted\_avg\_income,

sum(t7d\_cases) t7d\_cases,

sum(t7d\_cases) \* 1.0 / sum(total\_population) cases\_by\_population

from bottom\_25\_cases

group by date

)

select top\_25\_totaled.date,

top\_25\_totaled.cases\_by\_population cases\_by\_population\_top\_25\_income,

bottom\_25\_totaled.cases\_by\_population cases\_by\_population\_bottom\_25\_income,

top\_25\_totaled.cases\_by\_population - bottom\_25\_totaled.cases\_by\_population perc\_delta

from top\_25\_totaled

join bottom\_25\_totaled

on top\_25\_totaled.date = bottom\_25\_totaled.date

Table

Description automatically generated

Table

Description automatically generated

--5. For ZCTA 85203, Arizona, what is the population broken out by age group?

-- How does that compare to the State?

--- Heres the answere by population but this doesn't really tell us how 85203 compares. We will have to normalize.

--- We can normalize this by taking the population of the repective location and divide by the total population.

--- ie. zcta age group population divide by zcta total population.

select zcta,

age\_group,

population zcta\_popualtion,

dbo.dc\_County\_AgeGroup(zcta,age\_group) county\_population,

dbo.dc\_State\_AgeGroup(zcta, age\_group) state\_population

from dc\_age\_group

where zcta = '85203'

order by sort\_key

-- Normalizing the value per the above.

--- ie. zcta age group population divide by zcta total population.

select zcta,

age\_group,

population \* 1.0/ sum(population) over () zcta\_perc,

dbo.dc\_County\_AgeGroup(zcta,age\_group) \* 1.0/ sum(dbo.dc\_County\_AgeGroup(zcta,age\_group)) over () county\_perc,

dbo.dc\_State\_AgeGroup(zcta, age\_group) \* 1.0/ sum(dbo.dc\_State\_AgeGroup(zcta, age\_group)) over () state\_perc

from dc\_age\_group

where zcta = '85203'

order by sort\_key

go

Table

Description automatically generated

-- 6. What Counties had the highest deaths by percentage of population?

-- The population must be over 1000 people.

-- Include their household median income.

with population\_counties as (

select county\_id,

county,

state,

sum(population) population,

sum(population \* household\_median\_income) / nullif(sum(household\_median\_income),0) household\_median\_income

from dc\_geo\_population

group by county\_id, county, state

)

select top 25

population\_counties.county\_id,

population\_counties.county,

population\_counties.state,

population\_counties.population,

population\_counties.household\_median\_income,

total\_deaths,

total\_deaths \* 1.0 / population\_counties.population death\_perc

from population\_counties

join dc\_covid\_county

on population\_counties.county\_id = dc\_covid\_county.county\_id

where population >= 10000

and population\_counties.county\_id != 36061

order by death\_perc desc

Table

Description automatically generated

---7. Group the age groups that are considered the high-risk age categories for COVID. That is age groups over 65 and under 5.

--- Make this a calculated field that is a Boolean field being either 1 or 0; high risk or low risk.

--- Total the population by high risk and low risk by county.

--- Any counties that are greater than 30% being high risk what is the number of COVID cases and deaths by total population vs low risk counties.

--- Group these counties into 2 rows, high risk and low risk. How do they total cases and deaths compare to each other?

go

with risk\_counties as (

select dc\_age\_group.county,

covid\_age\_group,

SUM(dc\_age\_group.population) population\_age\_group,

coalesce(SUM(dc\_age\_group.population) \* 1.0 /

nullif(SUM(SUM(dc\_age\_group.population)) over (Partition by dc\_age\_group.county),0),0) perc\_population

from dc\_age\_group

group by dc\_age\_group.county,

covid\_age\_group),

county\_population as (select county\_id,

county,

sum(population) total\_population

from dc\_geo\_population

group by county\_id,

county

)

select 'High Risk' as risk\_type,

sum(total\_population) total\_population,

sum(total\_cases) \* 1.0 / sum(total\_population) perc\_cases,

sum(total\_deaths) \* 1.0 / sum(total\_population) perc\_deaths

from county\_population

left join dc\_covid\_county

on county\_population.county\_id = dc\_covid\_county.county\_id

where county\_population.county in (select distinct county from risk\_counties where perc\_population > 0.30 and covid\_age\_group = 'High Risk Age Group')

union

select 'Low Risk' as risk\_type,

sum(total\_population) total\_population,

sum(total\_cases) \* 1.0 / sum(total\_population) perc\_cases,

sum(total\_deaths) \* 1.0 / sum(total\_population) perc\_deaths

from county\_population

left join dc\_covid\_county

on county\_population.county\_id = dc\_covid\_county.county\_id

where county\_population.county in (select distinct county from risk\_counties where perc\_population > 0.90 and covid\_age\_group = 'Low Risk Age Group')

Table

Description automatically generated

-- 8. How many Deaths Vs. Total Population

go

with county\_population as (

select county\_id,

sum(population) population

from dc\_geo\_population

group by county\_id

)

select SUM(total\_deaths) total\_deaths,

SUM(population) total\_population,

SUM(total\_deaths) \* 1.0 / SUM(population) death\_perc

from dc\_covid\_county

join county\_population

on dc\_covid\_county.county\_id = county\_population.county\_id



### Data Visualizations:

Population by ZCTA and Age Group

Chart, histogram

Description automatically generated

Population by State and Age Group

Chart, histogram

Description automatically generated

Daily COVID Cases for Maricopa County

Chart, line chart

Description automatically generated

Daily COVID Deaths for Maricopa County

Chart, histogram

Description automatically generated

Daily COVID Deaths per cases for Maricopa County

Chart, histogram

Description automatically generated

Top 25 Household Median Income by County (Image 1) & Population (Image 2)

Chart, bar chart

Description automatically generated

The Population of Top 25 Household Median Income sorted by income.

Chart, bar chart

Description automatically generated

Bottom 25 Household Median Income by County (Image 1) & Population (Image 2)

Chart, bar chart

Description automatically generated

The Population of Top 25 Household Median Income sorted by income.

Chart, bar chart, histogram

Description automatically generated

### Reflection:

I have worked with databases now for 9 years and at first, I was under the impression that I would only get a little out of this class. In short, I was wrong. From Logical and Conceptual Models to Normal Form and then creating functions and procedures, I’m able to really approach database tasks and problems more efficiently and intelligently. For example, I would have skipped the modeling phase, dodged the normal form exercise and begin grabbing data to put into tables prior to this class. I would have ended up spending an exorbitant amount of time fixing my table, making mistakes when doing analytics, and had to update multiple tables with the same data. Something I could have done better is understanding the COVID data. What I discovered later on is that the cases and deaths where a cumulative number. I had to transform that data, drop the old and insert the new records after discovering this. How I built my database made it easy to perform this task. The COVID data was by county but broke it’s own rule by combining New York City Counties into one County called New York City. I could have created an additional table that stored the county codes associate to New York County, then joined up on this table to bring back to the census data. The last thing I could have done better, which I started to do, was create more views. For example, I created a zcta that had county, state as the dimensions and household median income as the metrics. I also did this for age group. I should have created a few more views. One for Race and then an aggregated view on county and state. This view would have made it easy to perform analytics on. Rather than building long queries users can just query these tables.

### Summary:

In building this database I had an objective to use census data to help answer questions surrounding the impact of COVID on our society. The lenses I was exploring was from a socioeconomics, age, and race perspective. I thought by weighting cases and deaths by population I could discern what the impact looked like for each of these groups. I will say there’s probably a more intelligent way to discern this information and I’m excited to continue my data discovery exercise outside of this class. I’ll probably implement the views I discussed in my reflection to make the analysis easier to perform. Deploy functions to aggregate metrics in a more straightforward method. Grow my understanding of procedures and implement them to help manage my data. I cannot help to be thankful of everything I learned in this class and ready to apply my new skills.

# Project 2 – IST 718 Big Data Analytics

## Course Description

A broad introduction to analytical processing tools and techniques for information professionals. Students will develop a portfolio of resources, demonstrations, recipes, and examples of various analytical techniques. You will find if much easier to succeed if you have completed IST687, IST777 or both. Familiarity with command-line interfaces, basic quantitative skills, including statistics, as well as programming skills with languages such as R or Python. Most of the course work will be using Python, Spark, and Tensorflow.

## Learning Objectives:

1. **O**btain data and explain data structures and data elements.
2. **S**crub data by applying scripting methods, to include debugging, for data manipulation in Python, R or other languages.
3. **E**xplore data by analyzing using qualitative techniques including descriptive statistics, summarization, and visualizations.
4. **M**odel relationships between data using the appropriate analytical methodologies matched to the information and the needs of clients and users.
5. Interpret the data, model, analysis, and findings. Communicate the results in a meaningful way.
6. Select an applicable analytical methodology for real problems in areas such as business, science, and engineering.

## Deliverables(s) – Predicting Zillow House Prices

Looking to new markets to open a business or buy a home can be challenging. A way to solve this problem is to look to the housing market. This can really help in understanding the economic health of an area. In this exercise understanding historical returns, the risk of an area, and by using forecasting techniques a Data Scientist needs to distil where a business should be opened in United States.

## Project Process & Development

### Obtain

There were 4 datasets used in performing this analysis:

1. Zillow Static Data set found at <https://files.zillowstatic.com/research/public/Zip/Zip_Zhvi_SingleFamilyResidence.csv>
2. Zip Code Tabulation Area (ZCTA) Household Median Income and Population level data from the Census Bureau.
3. County Household Median Income and Population level data from the Census Bureau.
4. State Household Median Income and Population level data from the Census Bureau.
5. State code data mapping to plot on a geographic map.

### Zillow Data

To obtain the Zillow data set the Pandas read csv function was used by inserting the above url.

Table

Description automatically generated

### Census Data

To obtain the census data, the following code lines were used:

1. Ping the census bureau api for Household Median Income and Population by year
2. Append the data to a dataframe
3. Loop through 2011 – 2020, this is everything that the census bureau has.
4. Rename the headers to Median Income and Population
5. Create a Pandas Dataframe

A picture containing text

Description automatically generated

Table

Description automatically generated

This data frame shows the House Hold Median Income by year. We can clearly see that there is some data to be cleaned with every min is -66666, representing 0. Due to this the average gets thrown off. The median Household income can be seen at the 50%. By 2019 this increased by $8k. If accurate this track with inflation. Meaning inflation on average is 2%, over the course of 10 years that comes out to roughly a little above 20%

Graphical user interface, application, table

Description automatically generated

Table

Description automatically generated

This process was repeated for both the County and State level.

The mapping for State Name and State abbreviation for to map the geographic visualizations were manually inserted.

Text

Description automatically generated

### Scrub

### Zillow Data

For the Zillow Data the following was needed to be scrubbed:

1. Zip code needed to have leading 0’s. Meaning a Zip Code is 5 digits with 0 at the front in some instances
2. The dataframe was melted to have the dates as rows instead of columns
3. The date name and the values were then called date\_zestimate and zestimate.
4. NaN were dropped from the dataset completely. Due to having so much data losing about 25% wasn’t a huge hit like normal datasets.

Before

Text, table

Description automatically generated with medium confidence

After

Graphical user interface, text, application

Description automatically generated

Dropped Data Results:

Graphical user interface, text, application

Description automatically generated

### Census Data

For the census data the following was needed to scrubbed:

1. The ZCTA data was pretty clean after obtaining the data.
2. The county data needed to be reformatted to create FIPS codes which is just the state code concatenated with the county code.

Table

Description automatically generated

### Merging Zillow and Census Data

Merged the Zillow data with the census data. This is joining Census onto Zillow. The Zip Code in Zillow is a copy right of the postal service and the ZCTA is a copyright of the census bureau. These two Zip Codes are different, so this is not a perfect match. There is a mapping file but for this analysis combining on the Zillow Zip Code will be sufficient due to only 2% of the Zip Codes were unable to be mapped. This is opportunity to improve the precision of the analysis.

Graphical user interface, text, application, email

Description automatically generated

An example of a Zip Code 85203

Graphical user interface, text, email

Description automatically generated

### Explore

### Arkansas Metro Time Series Plot

Chart, line chart

Description automatically generated

The graph above shows the 4 Metro areas Average Zestimate. In terms of growth Hot Springs appears to be the best with Little Rock being second best. This is hard to tell looking pearly out this graph. Also there is a lot of risk or volatility in all the Metro’s besides Searcy. Using some finance techniques lets look at the overall Return, Risk, and Return over Risk also known as Sharpe Ratio.

### Arkansas Metro Percentage Return (1997 – 2019)

Chart

Description automatically generated

This graph is a good visualization in volatility. Hot springs from 1996 until current has more volatility than the group with Searcy having the second most. Confirming our suspicions. Although Searcy appears to have done better after 2008.

Overall, from 1996 until the end of 2019 Overall Returns were as followed:

Text

Description automatically generated

The Higher the Sharpe Ratio is better. Meaning that for every Return an investor received they took on smaller risk compared to other investments. Hot Springs return overall was 71% but an investor had to take on seen in the below chart compared to Little Rock where very risk was needed.

Text

Description automatically generated

### Arkansas Metro Results (2010 – 2019)

Text

Description automatically generated

Before Hot Spring overall was the better choice from a historical perspective with the Return and Risk balanced very well. Looking to just this past decade Searcy has better return and a much higher Sharpe Ratio. The Riskiest area is actually Hot Springs now with Little over much smaller risk. Looking at more recent data will be important in the modeling stage of the analysis.

Looking at 2010 until 2019 the results are different. For the remaining of the analysis.

### Arkansas Metro Population and Household Median Income (2010 – 2019)

Chart, bar chart

Description automatically generated

The population represented hear is just the zip codes that Zillow provided. This might not be a full representation of the population growth. There appears to be a disconnect with Searcy and Zillow data. Meaning that housing prices are going up, yet population increased the first 4 years and then is on the decline. Either this population is now becoming homeowners or houses are disappearing from the market. Census data is not always accurate and with the 2020 census data coming next year Searcy might see an adjusted population growth.

Chart, bar chart

Description automatically generated

Confirming the same data issue with Searcy shows that household median income isn’t increases although housing prices and population are. A big disconnect. The other graphs track with steady growth.

### Bonus Geographic Visualization –Median Housing Price

A picture containing text

Description automatically generated

### Bonus Geographic Visualization – Population

Map

Description automatically generated

A picture containing text

Description automatically generated

### Bonus Geographic Visualization - Household Median Income

Map

Description automatically generated

### Modeling

### Historic Risk and Return - USA

When looking to perform the modeling historic returns should be considered by Zip code. The chart below shows all Zip Codes color coded by State. The objective in the analysis is too have the most return with lower risk. Meaning If a zip code achieves 12% return and a Standard Deviation (Risk) of 2% and another zip code achieves 12% return with a Risk percentage of 1% then taking the ladder zip code is the most optimal solution. This can also be described as a Sharpe Ratio where return is divided by risk. The higher the Sharpe Ratio means a more balance Return over Risk solution.

Chart, scatter chart

Description automatically generated

This graph shows every state from 2010 – 2019. The size is represented the Sharpe Ratio. As can be seen that California has a higher return with lower risk than many of the zip codes across the nation. Let’s focus though on Arkansas.

### Historic Risk and Return - Arkansas

Chart, scatter chart

Description automatically generated

Historically Speaking zip code 72447 has the highest return at 5.5% with some of the highest risk. Depending on the risk tolerance of an investor will depend if investing in this zip makes sense.

The next 3 highest zip codes have similar returns at 5% but have different risk levels:

* 71740
* 72675
* 72645

The better of the 3 and of even the highest would be 72645 achieving 4.7% return with much lower risk than the highest.

Chart, scatter chart

Description automatically generated

KS = Blue

AR = Red

OK = Yellow

TN = Orange

When looking at Neighboring states, excluding Texas There is better returns over risk then Arkansas. Meaning any different color bubble than red and is above red means that there is more return for the same amount risk.

### Forecasting Arkansas Return - Scrubing

When forecasting for Arkansas in 2020 and finding the 3 highest Zip codes Facebook Prophet was used in modeling these results. Before running the prophet additional scrubbing had to be done:

1. Filtered for just Zestimates from 2010 – 2019.
2. Only looked at Zipcodes with historic returns greater than 3% over the past 4 years.
3. Why looking over 3% is because inflation on average is 2% and investors minimum need to be compensated for taking on housing risk by a factor of 1%.
4. Then looked at just Arkansas which comes out to be 420 data points.
5. Changed names so that the Prophet would ingest the data.
6. Date\_zestimate changed to ds and zestimate to y
7. Looped the data through prophet by zipcode and appened to an empty dataframe including the zip code as a column.
8. Found the annualized returns predicted, annual risk predicted, and the sharpe ratio predicted.
9. Please see the below code for these steps.

Graphical user interface, text, application, email

Description automatically generated

Table

Description automatically generated

Table

Description automatically generated

Table

Description automatically generated

### Forecasting Arkansas Return - Results

The below graphs shows that the 3 highest return for 2020:

1. 72630 – Diamond City with a return of 17% and risk of 0.9%
2. 72431 – Grubbs with a return of 16% and risk of 0.8%
3. 71935 – Caddo Gap with a return of 13% and risk of 0.5%

Of the 3 that are the highest in graph 3 – 5 the predictive power illustrates the error bands in this forecast. Zip code 71935 appears to have much better predictive power of the 3 where the other’s do not.

Table

Description automatically generated

Chart, scatter chart

Description automatically generated

Zip code: 72630

Chart, line chart, scatter chart

Description automatically generated

Zip code: 72431

Chart, line chart

Description automatically generated

Zip code: 71935

Chart, line chart

Description automatically generated

Table

Description automatically generated

# Project 3 – IST 664 Natural Language Processing

## Course Description

Linguistic and computational aspects of natural language processing technologies. Lectures, readings, and projects in the computational techniques required to perform all levels of linguistic processing of text.

This course is designed to develop an understanding of how natural language processing (NLP) can process written text and produce a linguistic analysis that can be used in other applications. This goal will be achieved by:

* Readings, lectures, and class discussions of the multiple levels of linguistic analysis required for a computer to accept natural language input, interpret it, and carry out a particular application.
* Lab exercises and assignments in using some of the computational techniques required to perform these levels of natural language processing of text.
* Studies of real­ world applications that incorporate substantive NLP modules.

The course primarily covers the techniques of NLP in the levels of linguistic analysis, going through tokenization, word­ level semantics, part ­of­ speech tagging, syntax, semantics, and on up to the discourse level. It also includes the use of the NLP techniques, such as information retrieval, question answering, sentiment analysis, summarization, and dialogue systems, in applications.

## Learning Objectives:

At the end of the course the student will be able to:

* Demonstrate the levels of linguistic analysis, the computational techniques used to understand text at each level, and what the challenges are for those techniques.
* Process text through the language levels using the resources of the Natural
* Language Toolkit (NLTK) and some rudimentary use of the programming language Python.
* Describe how NLP is used in many types of real­ world applications.

## Deliverables(s) – Classifying IMDB Movie Reviews

Can reviews be used to predict rating, or better yet can reviews be deconstructed using multiple Natural Language Processing techniques and then apply machine learning to predict a rating. For this project IMDB reviews were used across many different movies to see the predictive power in what people say.

## Project Process & Development

### Overall

The goal we set out today is to predict the rating of a list of comments using classification algorithms and different cleaning techniques. The exhaustive list of items includes tokenizing, filtering, pre-processing the word list. Then apply feature engineering techniques that span Unigrams, Bigrams, and all the way to subjectivity. Then use multiple models like Bayes Classifier, Random Forest, and Support Vector Machine Classification. To round off the analysis, we used cross-validation to help prove the results will help in the prediction task.

No stone was left un-turn, with 30 different permutations of accuracy scores created based on this analysis's filtering, feature engineering, and models. Let’s jump in and see how well we can apply a rating to a comment?

# 

### Reviewing the Data

The dataset we chose to review and analyze was the Kaggle movie review list. This dataset was produced for the Kaggle competition, which uses data from the sentiment analysis by Socher et al. The data was originally taken from the Pang and Lee movie review corpus based on reviews from Rotten Tomatoes website. Socher’s group used crowd-sourcing to manually annotate all the sub-phrases of sentences with sentiment label scoring. The ranges included the following: “negative”, “somewhat negative”, “neutral”, “somewhat positive”, “positive”.

Before reviewing the original dataset we needed to import all the necessary packages needed for preprocessing and filtering. First we created training and test data frames using the given train.tsv and test.tsv files that were given to us. We wanted to review exactly how many unique sentences and compare them to the total number of sentences supplied in the training dataset. The total number of full sentences (also considered “unique\_sentences”) were 8520, out of a total of 156,060 listed phrases (also considered “total\_sentence”). That is approximately 18 times the amount of unique sentences. Next, we assign a “FullSentenceId” to each phrase, while grouping them by the SentenceId, this way we can see which smaller phrases are assigned to which completed full sentence. After putting the training data into a table with the new columns we could see that this dataframe was similar to a break down tree, similar to the grammar context previously learned in the course. To better review this discovery, we created a phrase histogram that compared the count along the y-axis and number of phrases along x-axis. The number exceeded above 350 count as its peak early on at approximately 15 phrases as you can see below.

Chart, histogram

Description automatically generated

To continue our visualization analysis of the training data, we then compared the number of phrases to the sentiment scores that were assigned to them. As scores 1 and 4 were the least, the score of 2 had the largest upper and lower bounds with exceeding whiskers past the 50 count of assigned phrases as you can see below. This essentially confirmed that there was a lot of cleaning and filtering that needed to be done prior to creating a new model to run our experiments on.

Chart, box and whisker chart

Description automatically generated

### Tokenized data

In order to tokenize our data we first join the training data frame creating the “Phrase” column into ‘characters.’ Using a nltk.word function that tokenizes the testtext vector to make our tokens. And to make sure all of our characters are seen the same, we made sure they were all lowercase into a new vector named testwords. The length of testtokens outputted 981,478 total words without filtering. We then wanted to dive deeper reviewing our bag (“test words”) of words by using the FreqDist function. 25 of the most common words based on the frequency were displayed to see the depth of filtering that needs to be done.

A picture containing text

Description automatically generated

As you can see below, one of the top results was a comma and other “words” such as “‘s” or “n’t” were considered in that top 25 of most common. Therefore, filtering needs to be done in order to get a better accuracy of our dataframe. Below is a better visualization of the word frequency without filtering:

Chart, bar chart

Description automatically generated

## 

### Filtered and tokenized data

First step we decided to do was remove punctuations from our dataframe using the list(puncuation) function and apply that to the testwords vector we already had stored. The length of words decreased to 9,30730. The frequency distribution of our testwords using the 25 most common were compared cause a slight change. Next, we remove any non-alphabetical characters in our test words with Regular Expression practices by using the re.compile('^[^a-z]+$') function within defining an alpha\_filter. Doing this significantly causes 10% of our test words to decrease. However, we still need to remove periods as well. We repeat this re.compile('\ |\?|\.|\!|\/|\;|\:') function toward the newly created alphatestwords vector. This also decreases our number of testwords again.

### Pre-processed and tokenized data

Now that unnecessary characters have been removed from the training dataset, we have to focus on the words tokens themselves. A lot of the time, English stop words decrease accuracy or natural language processing so it is best to remove them to better our models. We obtained the list of English stop words from the NLTK corpus (total of 179). Then, we created a new vector of stoppedtestwords that removes the nltkstopwords from the alphatestwords. A drastic decrease in total words was applied here making a total of 568,580 words in the training dataset. Another frequency distribution was applied to the new total words vector. Some characters that are not considered valid words were displayed in the most common list. Taking another list of morestopwords taken from a previous lab was used in addition to other characters from the frequency distribution and then added to the nltkstopwords. Ultimately, this help increase the frequency distribution for a majority of the most common characters as you can see below:

Chart, bar chart

Description automatically generated

Another form of tokenizing data and pre-processing is stemming and lemmatizing words. This essentially is mering the words together in order to increase the frequency overall. For stemming we apply the porter.stem function toward the stoppedtestwords and join that with the total words vector. For lemmatizing we apply the word\_lemma.lemmatize function as well to the previously created stemmedtestwords. This exercise did not decrease the number of words but did help improve the frequency. That is due to words being combined together and therefore increased the occurrence of the words.

The picture below illustrates how stemming and lemmatizing moved the word film from 0.008 to 0.01.

Text

Description automatically generated

In summary the chart and graph below shows the effect of each filtering technique on that data. What can be seen is that filtering stop words nearly cut the total words in half with the other technique cutting about 8% from the data.

Graphical user interface

Description automatically generated with medium confidence

Chart, bar chart

Description automatically generated

**Pre-processed, filtered, and tokenized data**

The Final Step was to build a function that leverages the pre-processing and filtering techniques discussed above to then apply to each individual sentence. First, we defined a processkaggle function to read the kaggle training file, train and test a classifier. This incorporated a limiting argument, looping over lines, picking a random sample of length, and then printing the phrase data. A cleaning\_phrases function was then defined by creating a list of phrase documents that first tokenizes the list of words, then makes those tokens lowercase, removes punctuations, and removes any tokens that are less than 1 at length. The next function created was filter\_phrases which incorporated tokenizing again, removing non-alphabetical characters, removing words under three characters, removing all stopwords, and again removing any tokens that are less than 1 at length. Lastly, the stemming and lemmatize techniques were combined into one stemmatize\_phrase function which tokens the phrases again, stemming, lemmatizing, and returning a final token list. Below are screenshots of applying these condensed functions to our imdb dataset and printing out the final breakdown of these words.

**Text

Description automatically generated**

Output:

**['see a study in contrasts ; the wide range of one actor , and the limited range of a comedian', '2']**

**(['see', 'a', 'study', 'in', 'contrasts', 'the', 'wide', 'range', 'of', 'one', 'actor', 'and', 'the', 'limited', 'range', 'of', 'a', 'comedian'], 2)**

**(['see', 'study', 'contrasts', 'wide', 'range', 'one', 'actor', 'limited', 'range', 'comedian'], 2)**

**(['see', 'studi', 'contrast', 'wide', 'rang', 'one', 'actor', 'limit', 'rang', 'comedian'], 2)**

What can be seen here is that the first sentence is not tokenized. Each of these sentences have the rating at the far right. The next 3 in the output above shows the sentence tokenized with everything lowercase and punctuations removed. The third sentence filters any stop words from NLTK and the custom list created. The last sentence shows the Stemmatize and Lemmatize technique. This can be seen with the word “study” changing to “studi”.

### Feature Engineering

The objective for feature engineering is to create the featuresets by passing the sentences into a function that will add a true or false value for every unigram, bigram, Part of Speech etc. A lexicon of different features like part of speech, sentiment, or subjectivity can be applied. More of this will be discussed further in the document.

### Bag of Words & Unigrams Features

This type of feature tokenizing every word in the sentence applies a True or False rating. At the end of Unigram dictionary contains the sentiment. This can be seen in the chart below.

({'V\_film': False,

'V\_movi': False,

'V\_one': False,

'V\_stori': False,

'V\_make': False,

'V\_charact': False,

'V\_time': False,

'V\_good': False,

'V\_even': False,

'V\_work': False},

1)

To begin our feature engineering we first get all the words from our dataset imbd\_stemma and put it into a frequency distribution. The output of the printed length returns 7056. Out of that total, we take 1000 of the most common word items and create a word features vector. Next, we define a document\_features function that takes the keywords of a document for bag of words and unigram baseline. Imbd\_clean is stored back into a document vector to store so we can replace this instead of all the individual features. After creating a new dataframe called feturesets that includes the previously stored documents, keyword features and category features. Doing this increases the length up to 9998.

### Bigrams Features

A bigram group two words together in a phrase and then assigns a rating of True or False. For example “Dan is the man” would translate to “Dan is”, “is the”, “the man”. Then a chi squared measure is used to find the top 500 bigrams. The feature set will then include just these word phrases.

We used the nltk.collaction.BigramAssocMeasures function on all the words in our sequence and stored that into a finder vector. Next, we define the top 500 bigrams using the chi squared measure and store that into a bigram\_features vector. We then define a bigram\_document\_feature using both document, word\_features, and bigram\_features described beforehand. Using this function will create feature sets for all the sentences. To test these bigrams we use the train\_set again to test a classifier and report the accuracy at 45.6%

({'B\_10-year\_delay': False,

'B\_18-year-old\_mistress': False,

'B\_22-year-old\_girlfriend': False},1)

### Part-of-speech Features

A lexicon was used here to understand how the part of speech might play a role in a sentence and in a model. In the second picture below we can see the number of nouns, verbs, adjectives, and adverbs for the first sentence.

The POS feature takes a document list of words and returns a feature dictionary in order to run the default POS tagger (Standord tagger) on the document. This way the feature can count 4 types of POS tags to use as features, such as, nouns, verbs, adjectives, and adverbs. The length of POS\_features outputted 1004. Below is an example of the first sentence.

Graphical user interface, text

Description automatically generated

## 

### Sentiment Lexicon (LIWC)

This feature is also considered linguistic inquiry and word count. This form of text analysis essentially calculates the degree to which the various categories in our kaggle training dataset of words are used in text. This lexicon assigned a positive or negative rating then adds to either a pos or negative list in the dictionary. A sample of data can be seen in the jupyter notebook and is very similar to the other examples above.

Using the sentiment\_read\_LIWC\_pos\_neg\_words.read\_words function was stored into a path vector, which was then separated into a positive and negative path. We created a LIWC\_feature that includes word counts of the subjectivity words. The negative features will have a number of weakly negative words and 2 times the number of strongly negative words. The final length of LIWC\_featuresets outputs 9998.

### Subjectivity

To begin engineering subjectivity, we imported in the necessary readSubjectivity libraries from the Kaggle movie reviews. Once the SLpath was stored into a vector, we defined the features that included word counts of the subjectivity words plus 2 times the number of strongly positive and negative words. Keeping in mind that positive features have similar definitions, not counting the neutral words. If the word was tagged as “weaksubj” or “strongsubj” it would now be considered positive or negative. The final length of SL\_features outputted 1002.

# 

### Experiments

### Bayes Classifier

Once our condensed features were created, we decided to test out our training\_set and test\_set using naive Bayesian Classifier. After taking 500 random samples of featuresets within the train\_set and test\_set, we used the NLTK Naive Bayes Classifier function and evaluated the accuracy and it came out to be 53.8%. Using the bigram\_featuresets the accuracy resulted in 54.4%. Using the POS\_featuresets to train and test the classifier our accuracy increased slightly to 54.6%. Lastly, the subjectivity lexicon resulted in a 56.4% accuracy and sentiment LWIC featuresets resulted in 53.8%.

### Cross Validation

In the cross validation experiment we created a cross\_validation\_accuracy function that the number of folds and feature sets, then iterates over the folds. By using different sections for training and testing, this function will print the accuracy for each fold and the average accuracy at the end of the output. When using the unigram word features and using a number fold of 3 the mean accuracy resulted in 48.0% with each fold size being 3332. When using the bigram\_featuresets and the same number of folds, the mean accuracy also resulted in 52.1%. Using the POS\_featuresets within the cross\_validation\_accuracy function, the mean accuracy resulted in 53.9%. Lastly, using the subjectivity lexicon, the mean accuracy resulted in 53.1%.

### Sci-kit Learn - Advance Task

Sciki-Learn is another machine learning program used for word efficiency models like NLTK. However, it was used within our program to see if the applications and accuracy of our models were in comparison to our other classifier and feature combinations. The first classifier we used was a random forest using the POS\_featuresets and that accuracy result was higher than the NLTK with 57.8%. The next one we used was the Support Vector Machine classifier using the same POS\_featuresets and that resulted in 54.6%, similarly to the NLTK.

# 

### Results

After building 3 filters, 5 feature engineering, and 2 models ran every permutation of possible experiments. This included 30 different experiments between Bayes NLTK and Random Forest Scikit learn. The results ranged from 56% to 44% with each new experiment incrementally improving from the other. The overall results are still poor with accuracy scores equating to nearly 50/50 shot in predicting the right rating based on comments.

Chart, bar chart

Description automatically generated

Regardless, drilling down into the top 5 experiments which includes the following:

1. Bayes LWIC using the clean feature
2. Bayes Unigram Clean feature
3. Bayes Part of Speech clean feature
4. Scikit Random Forest Bigram Clean feature
5. Bayes Subjectivity clean data feature

The top 5 ranged from 56% to 52% from an accuracy score. With Scikit Random forest as the only non NLTK model.

Chart, bar chart

Description automatically generated

Graphical user interface, application

Description automatically generated

Taking the top 5 results and running them each through cross validation but with 3,600 sentences and 12 folds instead of the original 1000 and 3 folds shown above.

Graphical user interface, application

Description automatically generated

The results from the cross validation shows that the average Accuracy is different then what we saw with the test and train data set. Overall the results are about 4% less than before. The top Accuracy is now the NLTK Bayes Part of Speech with 52%. When looking at the standard deviation, 2.4% is much better than the second best accuracy score, Bayes Bigram Clean. Meaning that POS with 52.4% and a standard deviation of 2.5% is the most optimal from the list above.

Chart, box and whisker chart

Description automatically generated

Looking at the 12 folds of the cross-validation results above Bayes POS Clean has a much higher average and the best median value than the rest. What is interesting is that the Bayes Subjectivity Lexicon using Stigmatizing and Lematizing has the tightest box plot of the bunch but the worst in Accuracy scores.

### Final Thoughts

The path to predicting a rating based on comments was long and arduous. Through filtering, transforming the data, applying multiple feature engineering techniques, and then modeling the featuresets, a conclusion can be seen. The power to predict comes at a great price. Throughout the process we only achieved an accuracy score of around 55%. Though the journey was hard, the exercises learned were invaluable. From the numerous cleaning and transforming techniques to ingest into a model. To the excitement of seeing results. Then to use methods to help prove the results, by cross validating the predictions. Although only 3 models were explored, additional models can be applied, and more data could be used. At what cost depends on the machine and time necessary to continue striving for a higher accuracy score. In summary the techniques in this paper illustrate what is needed to succeed in modeling text data. In addition, it serves as a terrific blueprint to begin applying Natural Language Processing in the real world.