Linear Algebra and Regression For Predicting Diabetes Mortality and it's Possible Contributors

Paper for DataHacks 2020
By: Nathan Zhao, Miles Labrador, Dylan Nelson

Expected Influence of Each Weight on Diabetes Risk

	Model Direction	Expected	Correct?
Support Group	⊕	⊖	✗
Young Vax 1	⊕	⊕	✓
Elder Vax 1	⊖	⊖	✓
Young Vax 2	⊕	⊕	✓
Elder Vax 2	⊖	⊖	✓
Depression	⊖	⊖	✓
High Bloodpressure	⊖	⊖	✓
High Colesterol	⊖	⊕	✗

Mission: Using a linear regression model for multiple features, we try to see if different topics and their statistics can be used to model diabetes prevalence or mortality in each state. If so, then are the weights reasonable? We expect features like high blood pressure to influence each of those in a specific direction, and if our model is good, it should show that as well.

Abstract: Using groups of questions from different topics of the dataset and different states, we try to see how they can be used to predict different diabetes outcomes. Using LSR and linear algebra, we made different models for each question.

After our findings for questions specifically targeting diabetes patients showed to be biased, we moved to more general questions that scientifically made sense. Both models showed to predict what we expected intuitively, but our second model we have more confidence in since we were able to reduce a greater amount of bias from it

Section 1: Data Cleaning/Preprocessing

Before settling on a research direction, we explored the data and parsed each row to figure out if it contained usable and useful information and we found that many rows were rendered useless by null values or lack of quantity. For example we removed all Identification columns as they were convoluted and non-intuitive duplicates of the same data. We removed geolocation because it was the location of each states' capital, and therefore the same as identifying which state the data was from..

```
data = Table.read_table("U.S._Chronic_Disease_Indicators__CDI_.csv")\
.drop("StratificationCategoryID3", "StratificationID3","StratificationCategory3", "Stratification3", "Response",\
      "StratificationCategoryID1","StratificationID1","StratificationCategoryID2","StratificationID2","DataValueTypeID",\
      "QuestionID","TopicID","LocationID","ResponseID", "DataSource","GeoLocation","StratificationCategory2","Stratification2",\
     "LocationAbbr","YearEnd")
```

```
ExampleTable = Table.read_table("U.S._Chronic_Disease_Indicators__CDI_.csv")
```

```
ExampleTable1 = ExampleTable.group("StratificationCategoryID3")
```

```
ExampleTable1
```

StratificationCategoryID3	count
	79188
nan	440530

```
ExampleTable2 = ExampleTable.group("Response")
```

```
ExampleTable2
```

Response	count
	79188
nan	440530

Left: All the different kinds of data values in the 'StratificationCategoryID3' and 'Response' columns, mostly null values. Removing the nulls was key to finding out what data can actually be analyzed.

From here we wanted to to find what data had the most to work with, we grouped by topic and found that diabetes was one of the most heavily surveyed datasets. From there we checked to see if the data was *consistent* (recorded every year) and *generalizable* by including all the states, it did.

-Notice only odd years had constant higher amounts of surveys (660), we think they only test every 2 years for some surveys, so we went with only odd years

```
diabetesTable = ExampleTable.where("Topic", are.equal_to("Diabetes"))\
.where("StratificationCategory1", are.equal_to("Overall"))\
.where("DataValueType", are.equal_to("Age-adjusted Prevalence"))\
.group("YearStart")
```

```
diabetesTable
```

YearStart	count
2011	660
2012	605
2013	660
2014	605
2015	660
2016	605

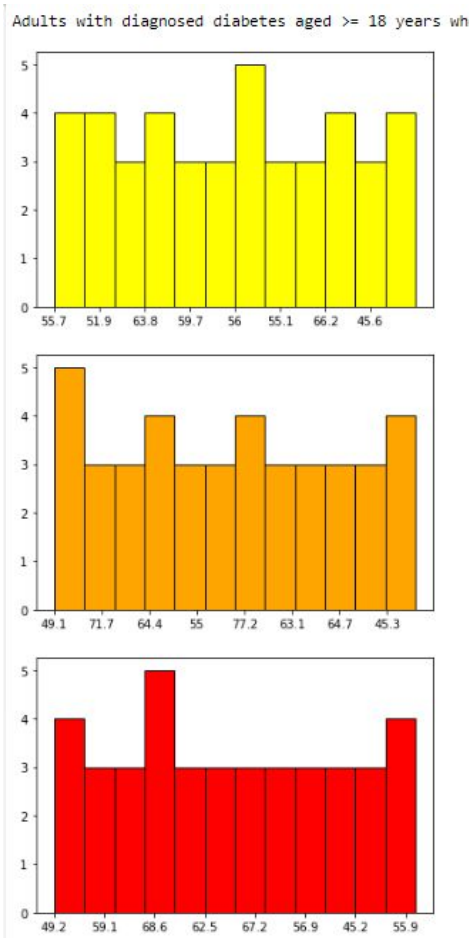
```
def string_to_float(column_val):
    try:
        converted_val = float(column_val)
    except:
        print("Error")
    return converted_val
# Changes the string values in the row we need to floats
ValidDT_converted = ValidDT.with_columns("DataValue", \
                                         ValidDT.apply(string_to_float, "DataValue"))
```

Other cleaning was done, like when we found out the data was in strings not floats

And our second data set we used had state names in abbreviations only, but we were using full names. And also it's reports were not proportions

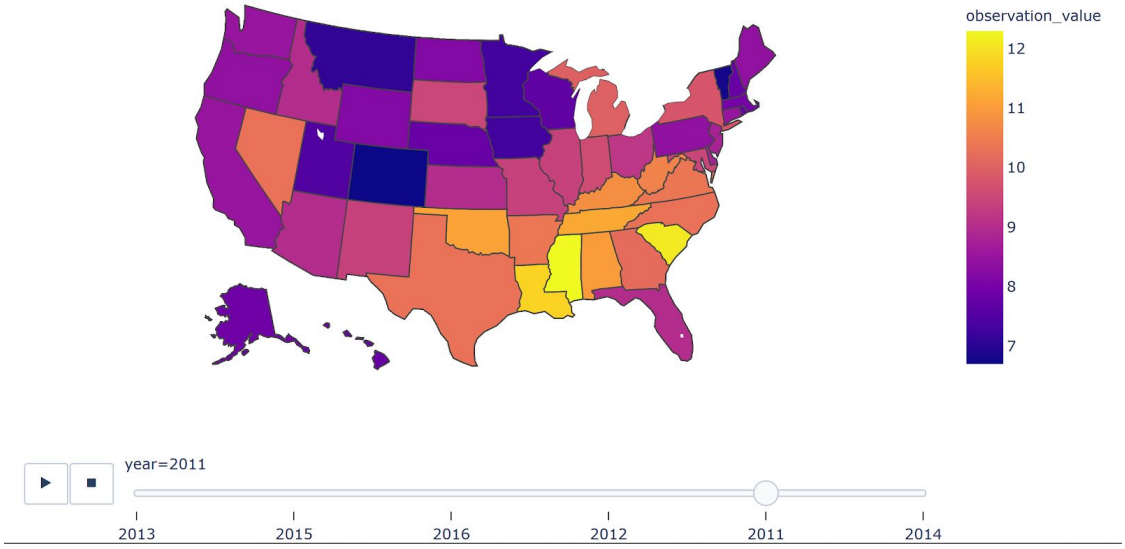
```
statesDict = {
        'AK': 'Alaska','AL': 'Alabama','AR': 'Arkansas','AS': 'American Samoa','AZ': 'Arizona','CA': '
        'CT': 'Connecticut','DC': 'District of Columbia','DE': 'Delaware','FL': 'Florida','GA': 'Georg
        'HI': 'Hawaii','IA': 'Iowa','ID': 'Idaho','IL': 'Illinois','IN': 'Indiana','KS': 'Kansas','KY'
        'MA': 'Massachusetts','MD': 'Maryland','ME': 'Maine','MI': 'Michigan','MN': 'Minnesota','MO':
        'MP': 'Northern Mariana Islands','MS': 'Mississippi','MT': 'Montana','NA': 'National','NC': 'N
        'ND': 'North Dakota','NE': 'Nebraska','NH': 'New Hampshire','NJ': 'New Jersey',
        'NM': 'New Mexico','NV': 'Nevada','NY': 'New York','OH': 'Ohio','OK': 'Oklahoma','OR': 'Oregon
        'PR': 'Puerto Rico','RI': 'Rhode Island','SC': 'South Carolina','SD': 'South Dakota','TN': 'Te
        'UT': 'Utah','VA': 'Virginia','VI': 'Virgin Islands','VT': 'Vermont','WA': 'Washington','WI':
        'WV': 'West Virginia','WY': 'Wyoming'
}
#making yVector
def translateName(abrev):
    return statesDict[abrev]
stateDeaths = stateDeaths.with_columns("STATE", stateDeaths.apply(translateName, "STATE"))
def convertToRatio(rate):
    return rate/100_000
stateDeaths = stateDeaths.with_columns("PROPORTION", stateDeaths.apply(convertToRatio, "RATE"))
yVector = stateDeaths.where("YEAR", are.equal_to(2017)).column("PROPORTION")
```

Section 2: Data Visualization



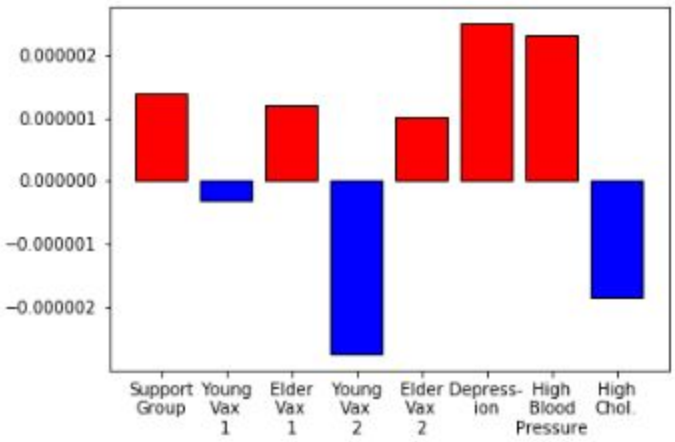
Before we finalized how we were going to compare states and predict, we started by seeing if we deal with the problem from a more probability focused approach. We made distributions of each question and their responses across the different states. We were expecting something more normal, but they were strangely uniform.

This is a distribution of the responses from the survey "Adults with diagnosed diabetes aged >= 18 years who have taken a diabetes self-management course". We chose to not go with this approach later on due to it not being as usable as normal distributions, and we ended up liking regression better because it is applicable to any dataset without having to worry about them also being normal

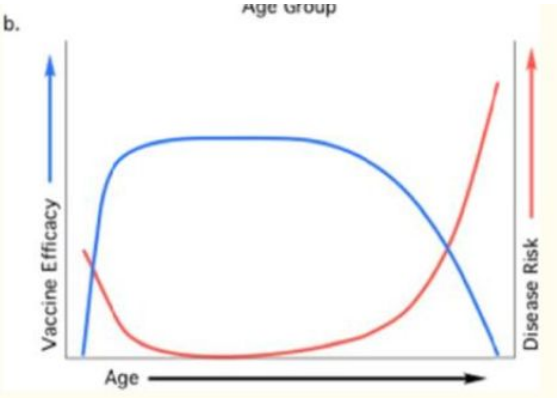


This choropleth map was built using plotly's data visualization module in python. Our first steps in our exploratory analysis led us to investigate diabetes in the US, as we found the topic had the most data points. This visualization maps and animates the percentage of overall prevalence of diagnosed diabetes among adults (aged 18 years or older) in each state over the span of multiple years. This visualization prompted us to consider the feasibility of considering how factors in states vary and may possibly contribute to a correlating diabetes diagnosed population.

After we did end up getting a prediction model for the first set of questions, we graphed the the weights and saw results that really went along with what we expected.



Here are the weights of each feature, and how much they are expected to contribute to diabetes mortality in the state. All of these lined up with what we predicted aside from the last and first bars. Our predictions required us to research each category, and we estimated a positive or negative influence for each [See Table 1]

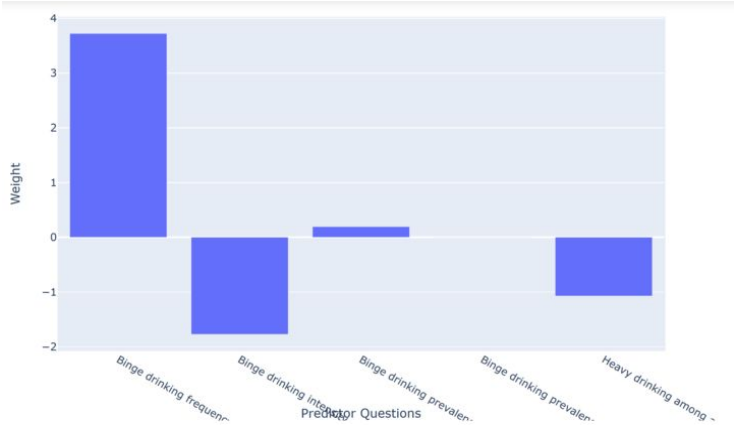


For example we figured vaccinations among young patients would correlate to a decrease in diabetes mortality, while depression and high blood pressure would correlate with an increase. This looked very reliable compared to what we expected, but still seemed a bit biased since these questions were targeted towards people who were expected to have, or had, diabetes. This only became more clear later after delving further into the data.

Note that in our distribution we noted that vaccination among elderly was detrimental, this came from a few studies observing the effects of *vaccine efficacy*, which states that as age increases, the risk eventually grows exponentially to outweigh the benefits

Figure and research from: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3616444/

Our data so far provided useful and interesting insights so we set out to streamline our data pipelining process in an effort to more efficiently explore insights and try to see a bigger picture in our data. Through a set of functions, we created code that takes in a dataset with values we want to predict (e.g. prevalence of diabetes) and a dataset with multiple factors (predictors) that may correlate with our prediction (e.g. Obesity levels in the state and Exercise levels in state). Our code then uses the states as a common data value to match our predictors' values to our actual prediction values. After this rudimentary cleaning is done, our code plots our weights with their respective predictor questions (factors) in order to be able to lead us to investigate possible causal factors that may be useful in helping reduce chronic illnesses.



This is one interesting correlating factors that we came across when running our regression function. We can begin to investigate the full statistic of "Binge drinking frequency among adults above 18 who binge drink" and their role in increasing diabetes prevalence in a state.

Section 3: NLP Analysis

When considering how to analyze our dataset using NLP, we saw that the only column that NLP could be applied to is the "Question" column. We first tried to use sentiment analysis, with our reasoning being that since the dataset was a result of a compilation of surveys across the United States, the sentiment rating of a question could perhaps correlate with the results of the question. To analyze sentiment, we used the SentimentIntensityAnalyzer object from the vader package in nltk:

```
from nltk.sentiment.vader import SentimentIntensityAnalyzer

sid = SentimentIntensityAnalyzer()
for sentence in data.loc[data.Topic == "Diabetes"].Question.unique().tolist():
    ss = sid.polarity_scores(sentence)
    print(sentence)
    print(ss)
    print()
```

This prints out each question, along with its sentiment values. Nltk's Vader processor takes each sentence, and calculates the sentiment of each word, and outputs the aggregate among the word scores.

In the sample above, we considered every unique question under the "Diabetes" topic and analyzed the sentiment of the question. However, we realized that due to the scientific nature of most of the words in the questions, the number of words that could have been considered emotional was small. In fact, most of the questions resulted in an overall neutral rating.

```
Foot examination among adults aged >= 18 years with diagnosed diabetes
{'neg': 0.0, 'neu': 1.0, 'pos': 0.0, 'compound': 0.0}

Mortality due to diabetes reported as any listed cause of death
{'neg': 0.281, 'neu': 0.719, 'pos': 0.0, 'compound': -0.5994}

Mortality with diabetic ketoacidosis reported as any listed cause of death
{'neg': 0.281, 'neu': 0.719, 'pos': 0.0, 'compound': -0.5994}

Influenza vaccination among noninstitutionalized adults aged 18-64 years with diagnosed diabetes
{'neg': 0.0, 'neu': 1.0, 'pos': 0.0, 'compound': 0.0}

Diabetes prevalence among women aged 18-44 years
{'neg': 0.0, 'neu': 1.0, 'pos': 0.0, 'compound': 0.0}

Adults with diagnosed diabetes aged >= 18 years who have taken a diabetes self-management course
{'neg': 0.0, 'neu': 1.0, 'pos': 0.0, 'compound': 0.0}

Prevalence of diagnosed diabetes among adults aged >= 18 years
{'neg': 0.0, 'neu': 1.0, 'pos': 0.0, 'compound': 0.0}

Dilated eye examination among adults aged >= 18 years with diagnosed diabetes
{'neg': 0.0, 'neu': 1.0, 'pos': 0.0, 'compound': 0.0}

Prevalence of depressive disorders among adults aged >= 18 years with diagnosed diabetes
{'neg': 0.178, 'neu': 0.822, 'pos': 0.0, 'compound': -0.3818}

Glycosylated hemoglobin measurement among adults aged >= 18 years with diagnosed diabetes
{'neg': 0.0, 'neu': 1.0, 'pos': 0.0, 'compound': 0.0}

Pneumococcal vaccination among noninstitutionalized adults aged >= 65 years with diagnosed diabetes
{'neg': 0.0, 'neu': 1.0, 'pos': 0.0, 'compound': 0.0}
```

Although the majority of the questions resulted in an overall neutral score, we decided to not move forward using sentiment analysis. We first considered adding our own input words to the corpus that Vader uses to analyze sentiment, but we realized that doing so will highly skew the results. For example, by adding the term “diabetes” to the corpus and correlating it with a negative sentiment, we would effectively skew the majority of the questions under this topic towards an overall negative sentiment.

Our next attempt with NLP was using the questions to find key terms between topics. We achieved this via splitting each question into n-grams. We could then check how many different topics have the n-gram in a question regarding the n-gram.

In [212]:
```python
import re

def generate_ngrams(s, n):
    s = s.lower()
    s = re.sub(r'[^a-zA-Z0-9\s]', ' ', s)
    tokens = [token for token in s.split(" ") if token != ""]
    ngrams = zip(*[tokens[i:] for i in range(n)])
    return [" ".join(ngram) for ngram in ngrams]
```

We defined a generate_ngrams function, such that given a string (with space delimiters) and a value of n, we could generate the n-grams of the string. Below are examples of generating n-grams of various values of n:

In [281]:
```python
string = "Data hackathons are super cool!"
generate_ngrams(string, 2)
```

Out[281]:
```
['data hackathons', 'hackathons are', 'are super', 'super cool']
```

```
In [282]: string = "Data hackathons are super cool!"
          generate_ngrams(string, 3)

Out[282]: ['data hackathons are', 'hackathons are super', 'are super cool']
```

```
In [283]: string = "Data hackathons are super cool!"
          generate_ngrams(string, 1)

Out[283]: ['data', 'hackathons', 'are', 'super', 'cool']
```

By considering the various n-grams, we can then compute the topics each unique n-gram exists in. That is, we can output a dictionary with a key value mapping of our n-grams (as strings) to a list of topics (as a list of strings).

```
grams_dict = {}
questions = data.Question.unique().tolist()

for question in questions:
    grams = generate_ngrams(question, 2)
    topic = data.loc[data.Question == question].Topic.tolist()[0]
    for gram in grams:
        if gram in grams_dict and topic not in grams_dict[gram]:
            grams_dict[gram].append(topic)
        else:
            grams_dict[gram] = [topic]

grams_dict
```

This applies to all questions within the Question column of the dataset, so we can effectively see which topics incorporate terms of word length n in their questions. We see the head of the general results below when we set n = 2:

```
Out[213]: {'amount of': ['Alcohol', 'Tobacco'],
           'of alcohol': ['Alcohol'],
           'alcohol excise': ['Alcohol'],
           'excise tax': ['Alcohol', 'Tobacco'],
           'tax by': ['Alcohol'],
           'by beverage': ['Alcohol'],
           'beverage type': ['Alcohol'],
           'type wine': ['Alcohol'],
           'type beer': ['Alcohol'],
           'type distilled': ['Alcohol'],
           'distilled spirits': ['Alcohol'],
           'obesity among': ['Nutrition, Physical Activity, and Weight Status'],
           'among adults': ['Diabetes',
            'Nutrition, Physical Activity, and Weight Status'],
           'adults aged': ['Diabetes',
            'Nutrition, Physical Activity, and Weight Status'],
```

Yet, due to the unordered nature of dictionaries, we need to sort our results dictionary to get understandable results. Because we want the n-grams that exist the most number of different topics, we sort by prioritizing the length of the values. We do so as follows:

```
res = {}

for key in grams_dict:
    res[key] = len(grams_dict[key])

{k: v for k, v in sorted(res.items(), key=lambda item: item[1], reverse=True)}
```

For which, we can finally see the 2-grams that are used in the most different topics:

```
Out[284]: {'65 years': 5,
           'aged 65': 4,
           'before pregnancy': 4,
           'influenza vaccination': 3,
           'use among': 3,
           '64 years': 3,
           '18 64': 3,
           'amount of': 2,
           'excise tax': 2,
           'among adults': 2,
           'adults aged': 2,
           '18 years': 2,
           'medicare eligible': 2,
           'eligible persons': 2,
           'smoking among': 2,
           'vaccination among': 2,
           'among noninstitutionalized': 2,
```

We considering other values of n, we get the distribution of n = 1 and n = 3 grams as follows:

```
Out[286]: {'pregnancy': 6,
           '65': 5,
           'by': 4,
           'before': 4,
           'among': 3,
           'with': 3,
           'influenza': 3,
           'more': 3,
           '64': 3,
           'days': 3,
           'youth': 3,
           'amount': 2,
           'excise': 2,
           'tax': 2,
           'adults': 2,
           'aged': 2,
           'years': 2,
           'for': 2,
```

```
Out[288]: {'aged 65 years': 4,
           'influenza vaccination among': 3,
           'prevalence among adults': 3,
           'adults aged 65': 3,
           'aged 18 64': 3,
           '18 64 years': 3,
           'prevalence among women': 3,
           '65 years with': 3,
           '64 years who': 3,
           'among adults aged': 2,
           'adults aged 18': 2,
           'aged 18 years': 2,
           'among medicare eligible': 2,
           'medicare eligible persons': 2,
           'eligible persons aged': 2,
           'vaccination among noninstitutionalized': 2,
           'among noninstitutionalized adults': 2,
           'noninstitutionalized adults aged': 2,
```

We can see that in all n-grams for n = {1, 2, 3}, n-grams relating to old age and pregnancy are the n-grams that are translated across the most topics. We chose not to relate pregnancy and old age as research topics, as these factors are generally not avoidable, and the increased risks of disease when such factors into account is well known and thus, trivial. Using higher order n-grams does result in more specific phrases that do not correlate to old age or pregnancy specifically, but the number of topics that correspond to the n-gram decreases dramatically, which thus decreases the n-gram to topic list correlation. Therefore, we did not find any substantial nontrivial results from n-gram processing.

Instead of grouping by sentiment or n-grams, we chose to analyze questions in a specific topic such that the question had the most data with respect to the DataValues column. This way, although we may not be organizing the most prominent data in terms of weights of questions under some natural language metric, but for the questions we do analyze, we ensure that the data

corresponding to the data is wide reaching. Specifically, we chose questions that satisfy two parameters:

1. Every state has a non-null answer to the question (by having a non-null DataValue)
2. Every year has a non-null answer to the question (by having a non-null DataValue)

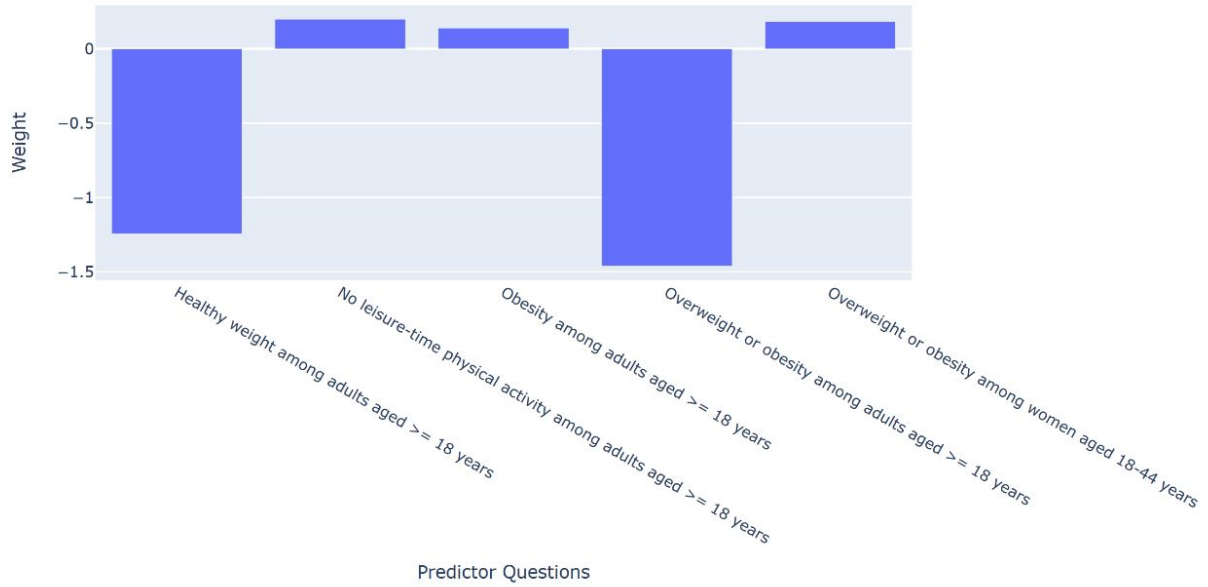
By taking the intersection of the questions that fall into these two categories, we effectively find the questions that have the most data relative to both the state responses (thus giving us data from a wide distance frame) and time response (thus giving us data from a wide time frame). For our data analysis we used data from such questions as indicating factors that lead to a positive or negative result of diabetes.

Section 4: Hypothesis Testing

For our hypothesis testing, we begin by analyzing a specific topic to run our model on. Since our linear regression model can run on any set of questions, we will narrow our general model onto one topic. For the sake of example, we will let the topic to run our model on to be Alcohol. We chose five questions under the Nutrition category that satisfy the results of our natural language processing, them being listed below:

- Heavy weight among adults >= 18 years
- No leisure-time physical activity among adults aged >= 18 years
- Obesity among adults aged >= 18 years
- Overweight or obesity among adults aged >= 18 years
- Overweight or obesity among women aged 18 - 44 years

Next, we set our null hypothesis to the statement that “Nutrition does not correlate to increased risk of diabetes”. By running our linear regression model on these five questions, we get the following results:



We see that two questions have significantly large weights in terms of magnitude, while the other three have weights of lower magnitude. Since our model states that negative weights correlate to less susceptibility of diabetes, we have that healthy weight and overweight adults >= 18 years leads to a lower susceptibility of diabetes. We see that our distributions between questions are not random, as there is a higher weight towards the negative values. Thus, we can say that there is a correlation between some of the question factors and diabetes.

Section 5: Proposal and Conclusion

In the end we used linear algebra to draw inferences about different chronic illnesses and their impact on diabetes mortality and prevalence. While our first conclusions looked very reliable, we chose to take them with a grain of salt due to further research in the surveying methods. From there we found ways to generalize our algorithm, and used it to see interesting new conclusions (like the potential effects of binge drinking on diabetes) and familiar observations supported by our study (like the benefits of vaccinations of certain diseases that do greater harm to diabetes patients). While we had to move from a few different models and methods of extracting meaning from the data, we were really satisfied with our results in the end, and thought we were able to express a lot from multi feature linear regression.

Additional Tables and links:

https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3616444/

- Where we got info on vaccinations in relation to diabetes

https://www.cdc.gov/nchs/pressroom/sosmap/diabetes_mortality/diabetes.htm

- Our second dataset

http://code.activestate.com/recipes/577305-python-dictionary-of-us-states-and-territories/

- Useful dictionary for translating from abbreviations

[Table 1]

Weights	Questions	ExectedInfluence	CorrectInfluence?	NLPExpectedInf
1.39228e-06	Adults with diagnosed diabetes aged >= 18 years who have ...	-	False	=
-3.12634e-07	Influenza vaccination among noninstitutionalized adults ...	-	True	=
1.19192e-06	Influenza vaccination among noninstitutionalized adults ...	+	True	=
-2.7273e-06	Pneumococcal vaccination among noninstitutionalized adul ...	-	True	=
1.00451e-06	Pneumococcal vaccination among noninstitutionalized adul ...	+	True	-
2.491e-06	Prevalence of depressive disorders among adults aged >= ...	+	True	=
2.31845e-06	Prevalence of high blood pressure among adults aged >= 1 ...	+	True	-
-1.82665e-06	Prevalence of high cholesterol among adults aged >= 18 y ...	+	False	=