

Capstone Project Presentation: Date-A-Scientist

Marshall Mamiya
Machine Learning Fundamentals
#cohort-oct-09-2018



Table of Context

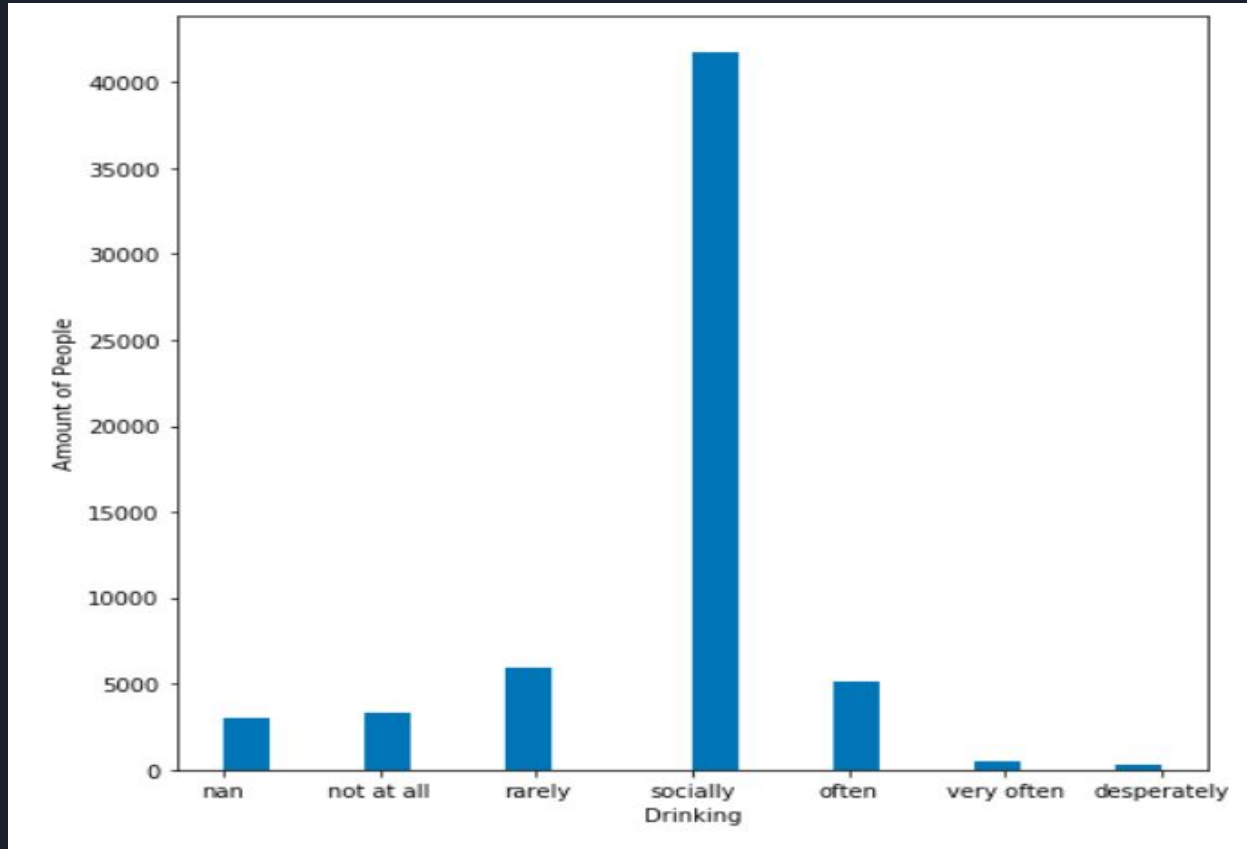
- Exploration of Data
- Visualization of Data (Graphs)
- Classification Approach (2)
- Regression Approach (2)
- Conclusion



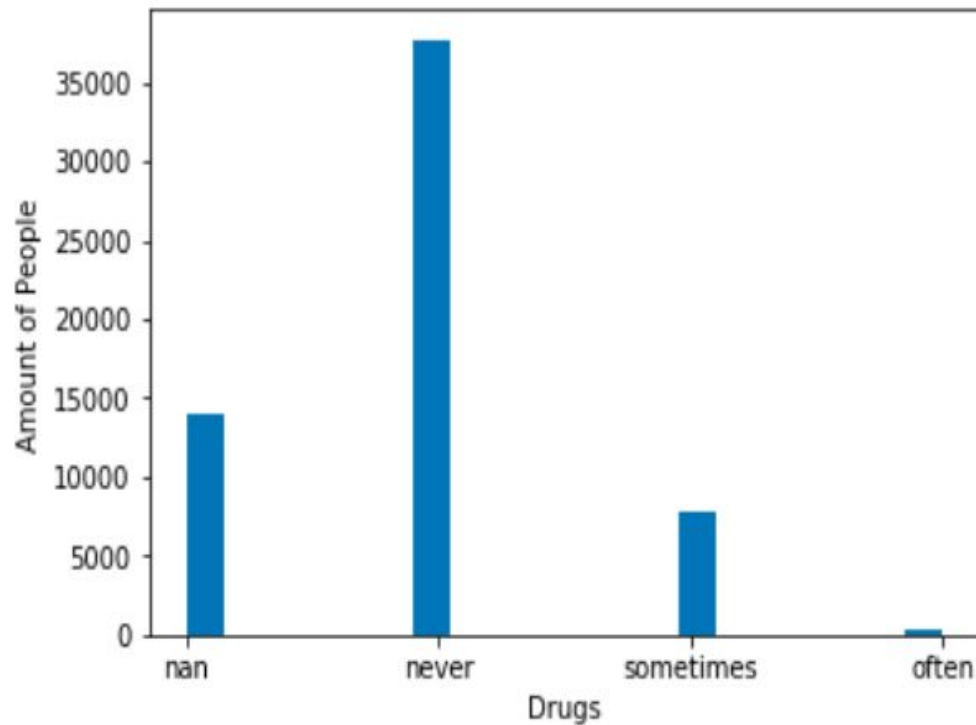
Exploration of Data

- Insights from exploring dataframe
 - Only 3 features have continuous numerical data (age, height, income)
 - Use regression models to predict those features above
 - Other features are categorical or text
 - Use classifier models to predict a categorical feature
 - Could use a countvectorizer on essay text for word count and then use Naive Bayes

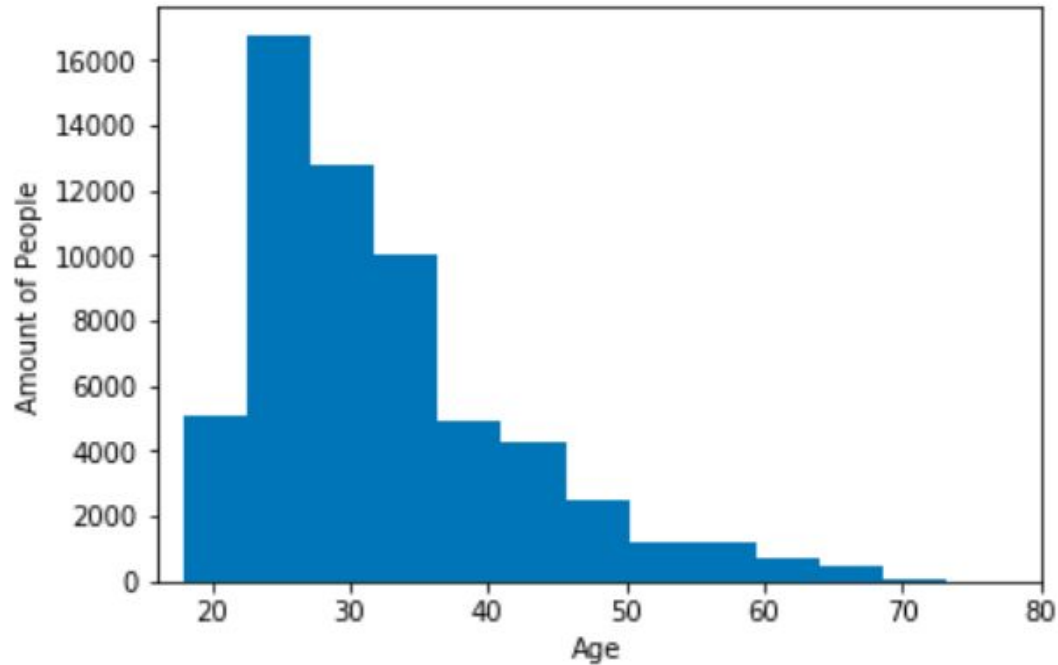
Visualization of Data: Graphs [1] : Drinking



Graph [2] : Drugs



Graph [3] : Age



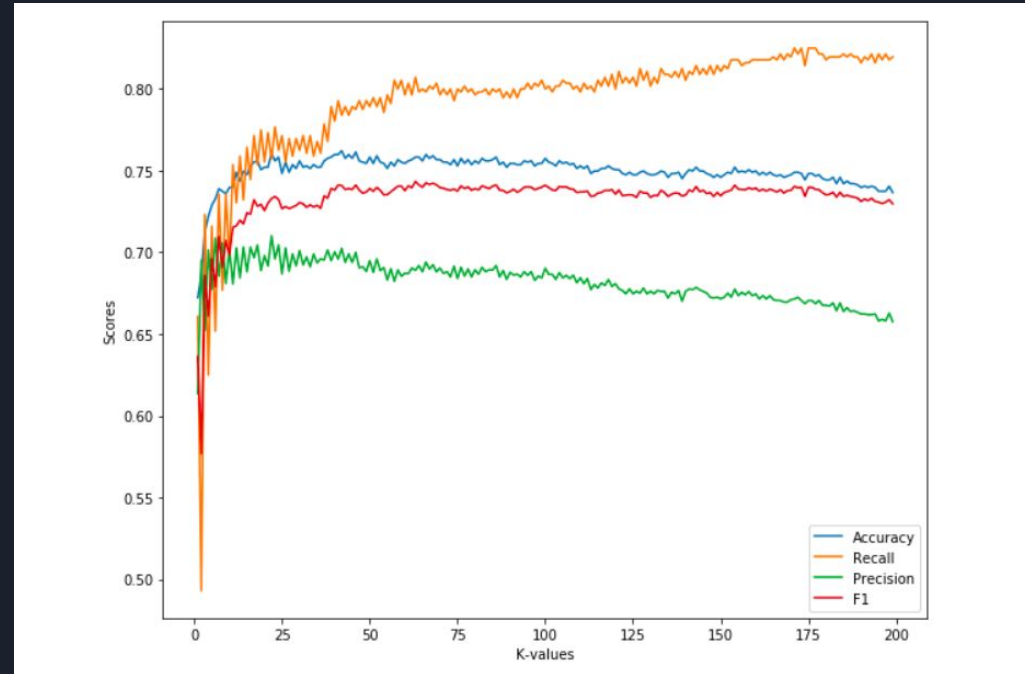


Classifiers: Problem One

- Problem One: Predict if somebody wants to have children or not using K-Neighbors and Naive Bayes
- Features used: [age, drugs, drinks, income, orientation, job, sex]
- Reason to use these features:
- These features are usually important factors to consider before having a child (e.g. having a high enough income to support child expenses)

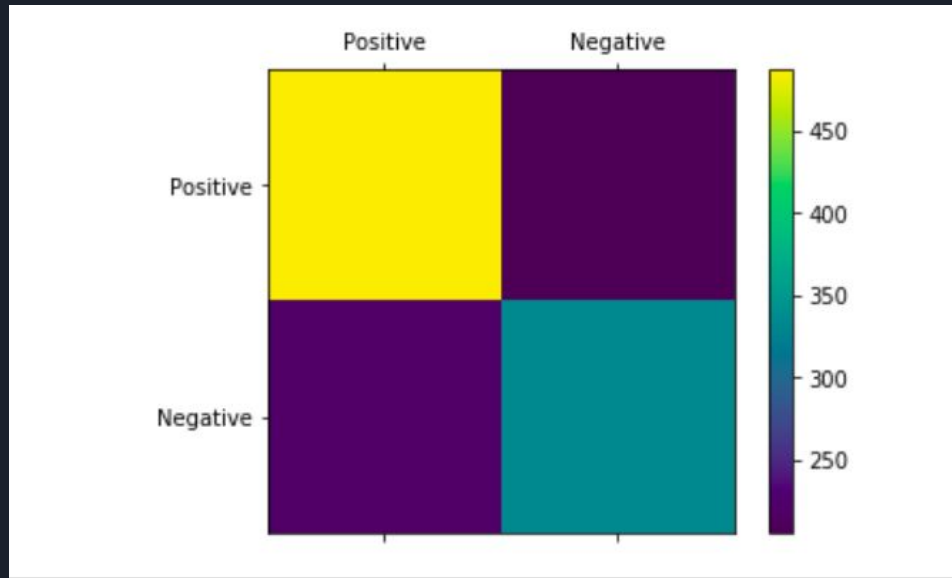
Classification Approach #1: K-Nearest Neighbors

- Total profiles used for model: 6236
- Time to run model: 20.8 ms \pm 2.31 ms per loop
- Time to run predictions: 95 ms \pm 1.1 ms per loop
- Dropped profiles with answers that don't state if they want kids or not in the offspring column (e.g. "might want kids", "have kids", etc.)
- Best scores (#K's, score) :
 - Accuracy: (42, .76)
 - Precision: (171, .83)
 - Recall: (22, .71)
 - F1: (63, .74)



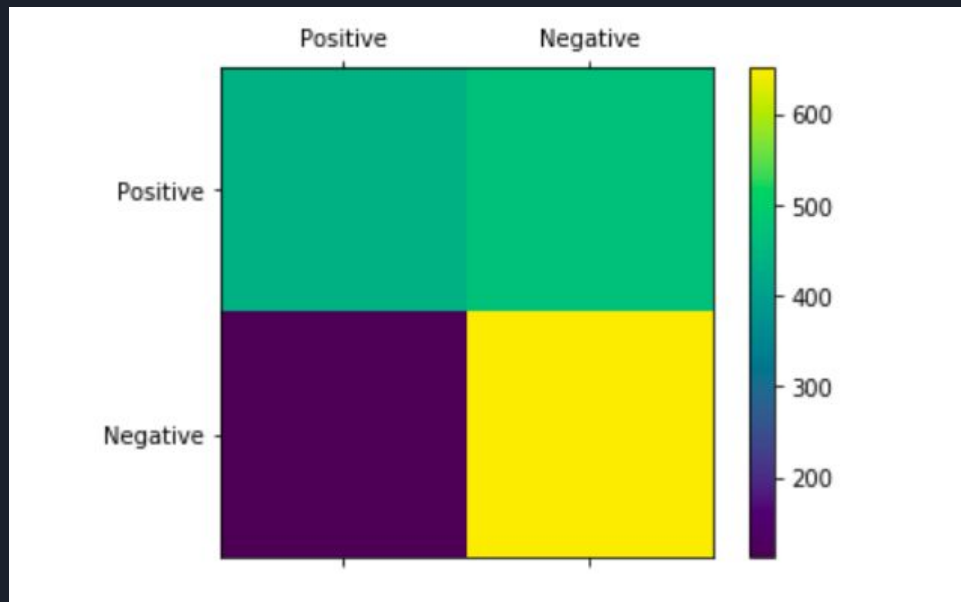
Classification Approach #2: Naive Bayes

- Total samples used for model: 6236
- Time to run model: 2.04 ms \pm 31.8 μ s per loop
- Time to run predictions: 121 μ s \pm 1.14 μ s per loop
- Dropped same amount of nans as KNN for the same reason
- Scores:
 - Accuracy = .66
 - Precision = .62
 - Recall = .60
 - F1 = .61



Classification Approach #3: Naive Bayes (Using Essays Only)

- Total samples used for model: 8383
- Time to run NB model (not including Countvectorizer): $8.16 \text{ ms} \pm 155 \text{ } \mu\text{s}$ per loop
- Time to run predictions: $1.89 \text{ ms} \pm 26.5 \text{ } \mu\text{s}$ per loop
- Filled nans with empty strings instead of 0 (since Countvectorizer doesn't take integers)
- Scores:
 - Accuracy = .65
 - Precision = .58
 - Recall = .85
 - F1 = .69





New Columns in Dataframe

- Since I used categorical features that contain strings, I mapped each categorical feature as a number in a new column
- Example:
- `smokes_map = {'0': 0,`
- `'no': 1,`
- `'sometimes': 2,`
- `'when drinking': 3,`
- `'yes': 4,`
- `'trying to quit': 5}`
- `df['smokes_code'] = df.smokes.map(smokes_map)`



Classifier Comparison

- Time to run model: The KNN model took the longest amount of time(18.76 ms longer than categorical NB). I was surprised that the Naive Bayes model with only essays took a shorter amount of time since it trained 2,000 more profiles than KNN model
- Time to run predictions: The Naive Bayes models won again in regards to the shortest runtime for predictions by a long run (KNN: 1.89 ms vs NB categorical: 121 μ s)
- Simplicity:
- KNN was the easiest model to work with because I had practice using KNN on Yelp data
- It was challenging to create an efficient function to determine the best essays to use Naive Bayes model on
- Scores:
- Since the consequences between false positives and false negatives are somewhat equal, the best model would be prioritized towards the best F1 score.
- Consequences are equal since..... nobody wants to find out later down the road that their potential soulmate doesn't have the same perspective on having kids :(
- Out of all classifiers, the NB model with only essay features had the best F1 score (.69), followed by KNN (.63) and lastly NB with categorical features (.61)



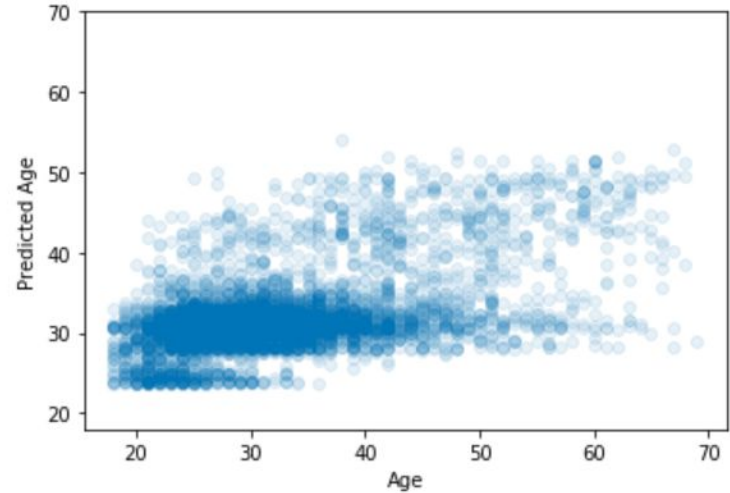
Regression: Problem Two

- Problem Two: Predict age using K-Neighbor regression and Linear regression models
- Features used: [job, sex, offspring, total_texting_num, total_older_words (description in slide 16)]
- Curious to see if the new columns I created have a stronger correlation than the given categorical features
- Want to get more practice using regular expressions in the process of creating the new columns

Regression Approach #1: KNN

- Time to run model: 164 ms \pm 26.5 ms per loop
- Time to run predictions: 114 ms \pm 2.99 ms per loop
- Best scores with #K = 40:
- MSE: 64.23
- R2: .28

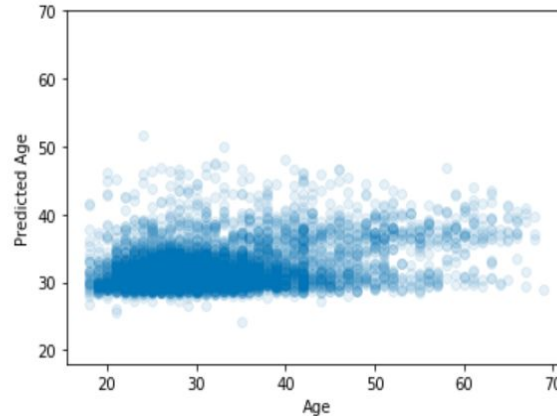
Mean Squared Error: 64.22589076384949
R2 Score: 0.2751987925487379




Regression Approach #2: Linear Regression

- Time to run model: 2.42 ms \pm 426 μ s per loop
- Time to run predictions: 111 μ s \pm 28.1 μ s per loop
- Scores:
- Train Score: 0.13
- Test Score: 0.14
- Mean Squared Error: 78.2
- R2: .18

```
Features Coefficients (most to least predictive): [('offspring_code', 1.118245830616006), ('total_older_words', 0.32645934541977517), ('total_texting_char', -0.2936655289636713), ('job_code', 0.23975351360954078)]  
Mean Squared Error: 78.20196092779348  
R2 Score: 0.11747622288448178
```





New Columns in Dataframe

`df['total_texting_num']` & `df['total_older_words']`

- `df['total_texting_num']`:
- Sum of all text emojis found in essays for each profile (example text emojis: :), :(, ;) etc.)
- `df['total_older_words']`:
- Sum of all words related to older people (e.g. divorced, son, daughter etc.) for each profile
- Process to create columns:
- Create a function that generates a duplicate dataframe to use regular expressions on
- Used regular expression to strip essays (alphabetical letters and HTML tags for “total_texting_num”) & (punctuations and HTML tags for “total_older_words”)
- Create function called `add_one` that counts the number of text emojis or “older words” within each essay
- New column takes the sum of text emojis or “older people words” for each profile
- Reasons for columns:
- For “total_texting_num”, curious to see if there is a negative correlation coefficient between text emojis used in essays and age.
- For “total_older_words”, curious to see if there is a positive correlation coefficient between “older people words” used in essays and age.



Regression Comparison

- Time to run model: KNN runtime was 162 ms longer than linear regression runtime
- Time to run predictions: KNN also had a significantly longer prediction runtime (KNN: 114 ms vs Linear: 111 μ s)
- Simplicity:
- Training both model was fairly straightforward
- Although the most time consuming and trickiest process was understanding how to use regular expressions to create the, “total_texting_num” and “total_older_words” column. Since I’m still very unfamiliar with regular expressions, it took a lot of trial and error to figure out the right pattern and syntax to use.
- Scores:
- Both models had a very low score that were under .30 (not impressed.... At all). Out of the two regression models, KNN produced best scores.
- Not surprising since all features had a weak linear relationship with the age feature so the results were likely to be unsuccessful.



Conclusion

- Answer to problems:
- With current features used for the training models, YES KNN and NB are reliable classifier to predict a profiles offspring choice between wanting kids or not
- With current features used for the training models, NO Linear and KNN are not reliable regressors to predict a profile's age
- Data and information that would possibly help answer problems:
- Profile pictures and information on how to create an age/image recognition model
- Next steps:
- Explore unsupervised learning and possibly incorporate a K-Means model that outputs an OkCupid match recommendation for profiles
- Continue to practice and learn more about using Pandas and possibly take a course in SQL
- Way down road, learn more about neural networks and the realm of Deep Learning