A Pythonic Job Search

Sam Raykhman

Github
LinkedIn

Job Searching CAN be Stressful

Objective: To establish a model that classifies jobs that have knowledge of Python as a requirement

Purpose: Help out myself and my peers to find a job for our qualifications

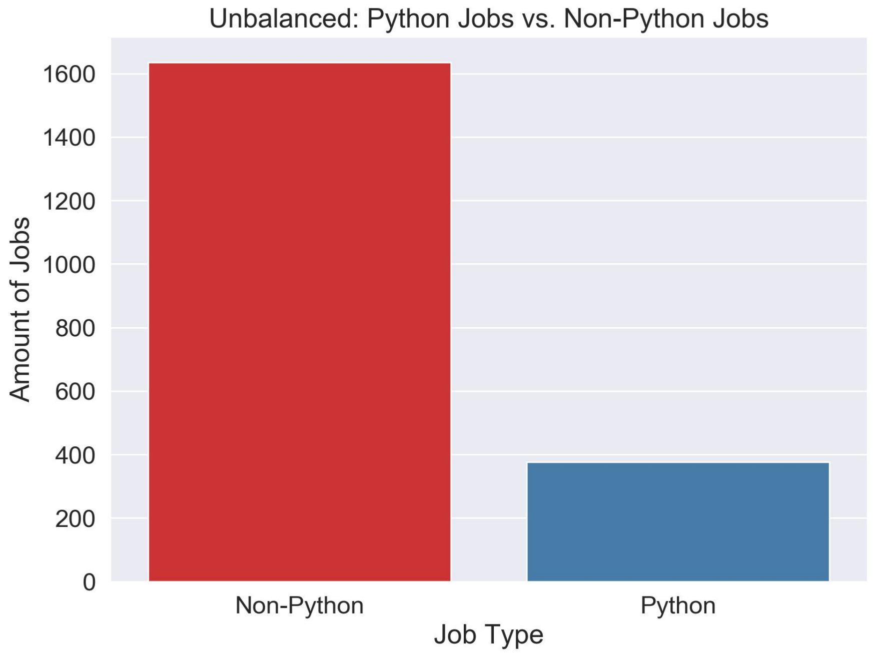
The Process

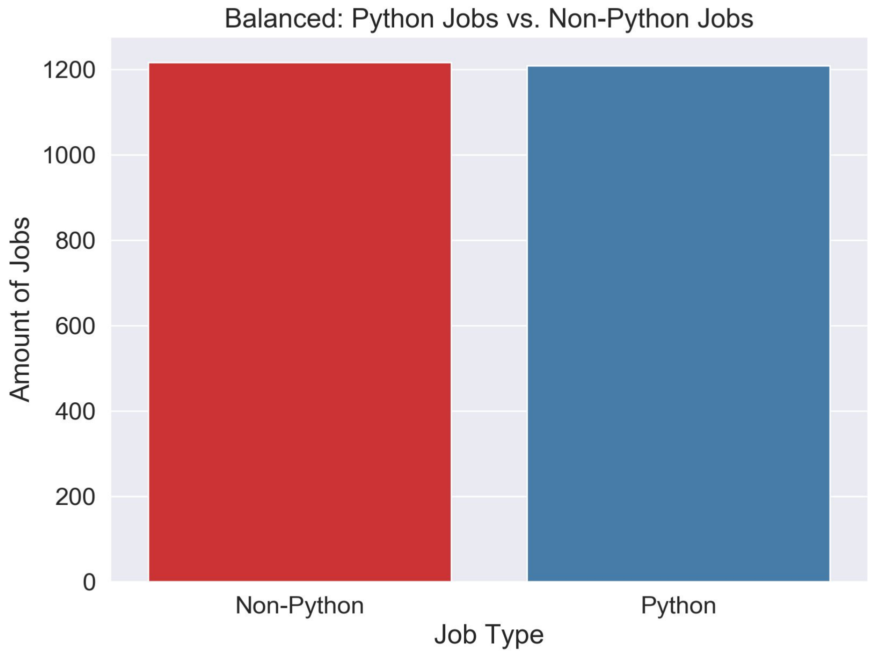
Data Collection	EDA	NLP Feature Engineering	Modeling	Evaluation
Web scraped my data from stackoverflow	Cursory examination of data	Tokenized, removed stop words and performed TF-IDF	Multinomial Naive Bayes, Random Forest, XGBoost	Do I need python or not?

Diving into the Data

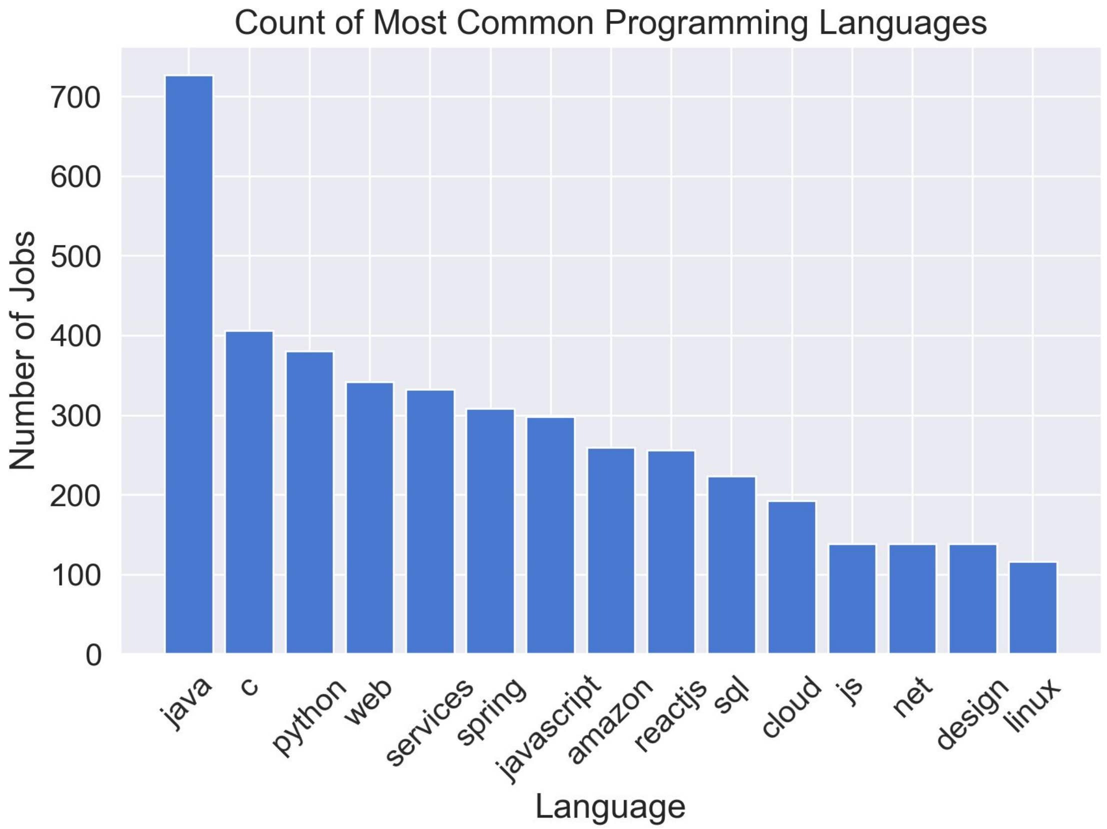
- I web scraped my data from stackoverflow’s job posting portal
    - At the time of my scraping there were 2893 jobs in total
    - After removing rows with empty descriptions or no list of programming languages, I was left with about 2100 rows.
- I created 5 features
    - Job Title
    - Description
    - Languages
    - Overview
    - URL
- Created target variable out of Languages column
    - If python was listed as a language required or not

Class Imbalance

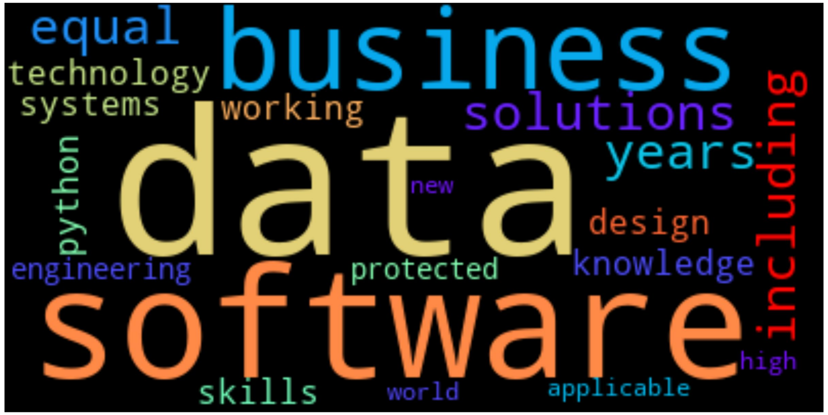




Common Languages

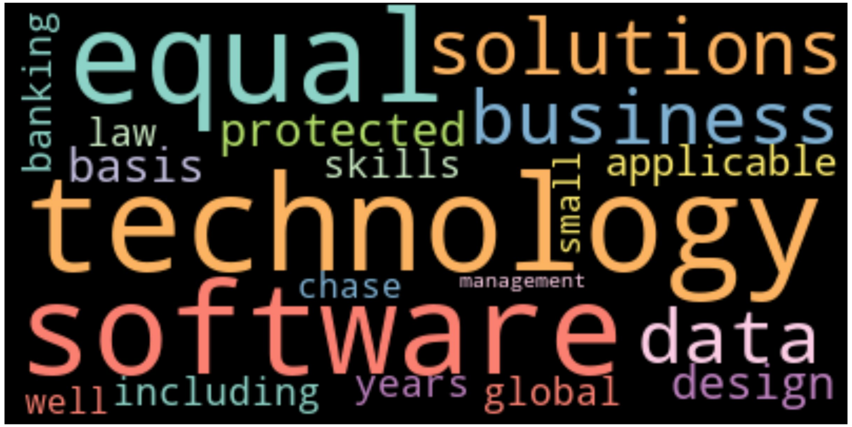


Word Clouds



20 most common words for jobs with Python as a requirement

20 most common words for jobs without Python as a requirement



Model Evaluation

Dummy

Accuracy: 0.8330

F1 Score: 0.0

- Strategy = most frequent

Naive Bayes

Accuracy: 0.8489

F1 Score: 0.4933

- Alpha = 0.0001

Random Forest

Accuracy: 0.8330

F1 Score: 0.0

- Criterion = gini
- Max depth = 1
- Max features = 1
- Max leaf nodes = 2
- # of estimators = 20

XGBoost

Accuracy: 0.8887

F1 Score: 0.6627

- Learning rate = 0.1
- Max depth = 3
- Min child weight = 1
- # of estimators = 200

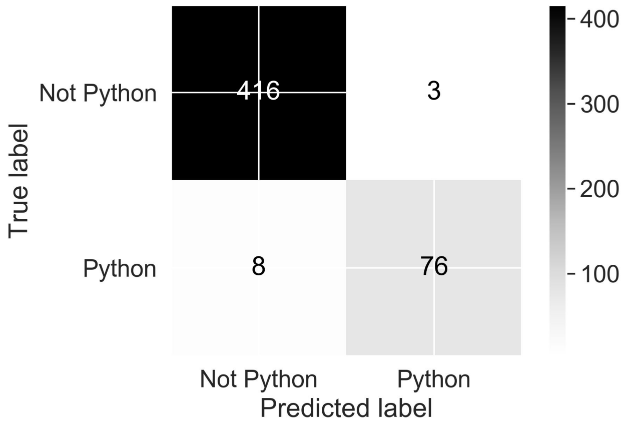
ADASYN XGBoost

Accuracy: 0.8429

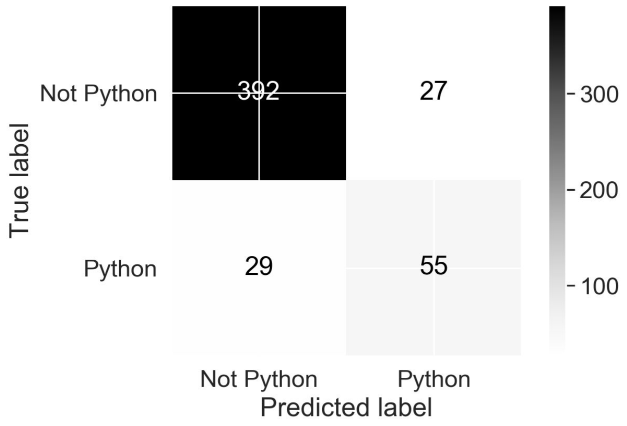
F1 Score: 0.6291

- Learning rate = 0.001
- Max depth = 5
- Min child weight = 0.01
- # of estimators = 100

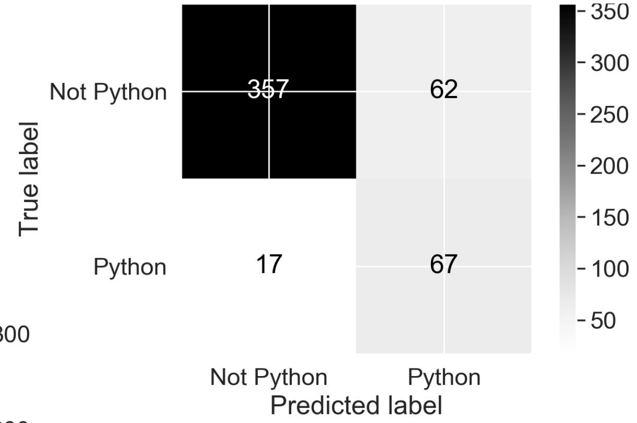
Confusion Matrices



Naive Bayes

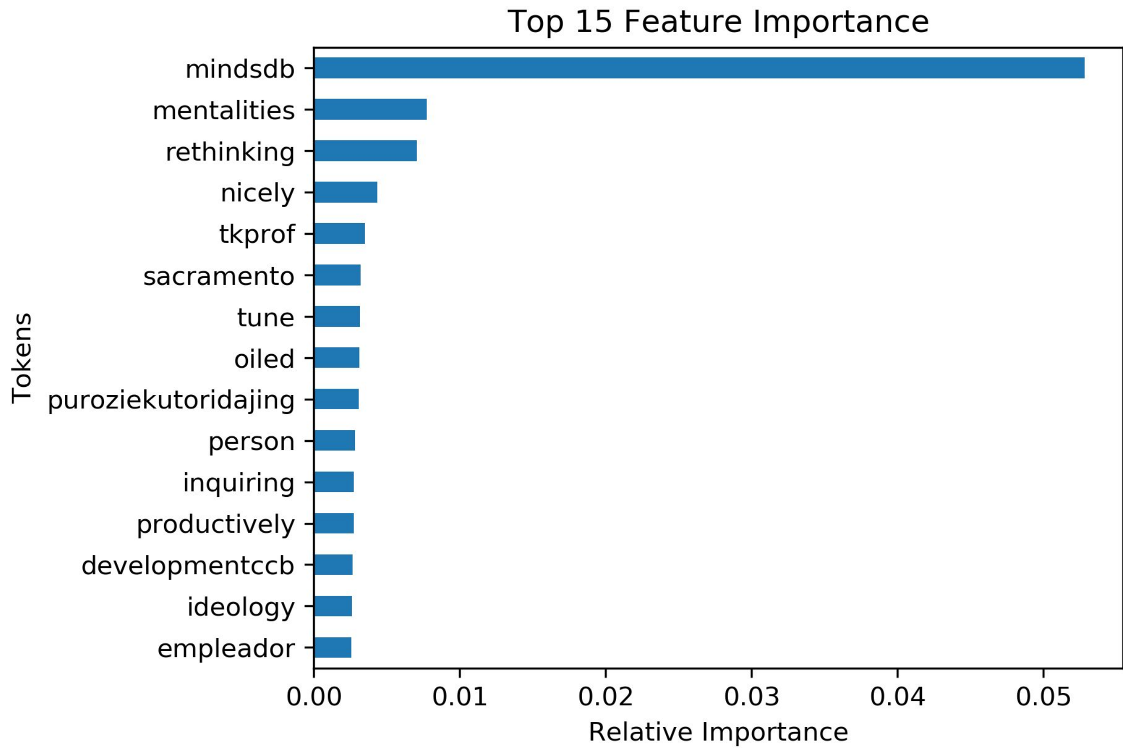


Unbalanced XGBoost



XGBoost with ADASYN

RF Feature Importance



Conclusion

In the real world, it can take hours to merely find a job that you qualify for and interests you. Let alone, the days and weeks of applying and going through the interview processes.

Using my unbalanced XGBoost model, I can easily filter the jobs to my (or other people's) qualifications, reducing the time and effort spent on the hunt; without hindering the results by generating false data points through SMOTE or ADASYN.

Future steps

- Incorporate N-grams, to check importance of phrases and not just words
- Use a Support Vector Machine Model, to attempt to increase the performance of my final model
- Create recommendation system
    - That would take qualifications, location, and some consumer preferences to return jobs in your area that match your requests

Thank you

Any Questions?

Github
LinkedIn