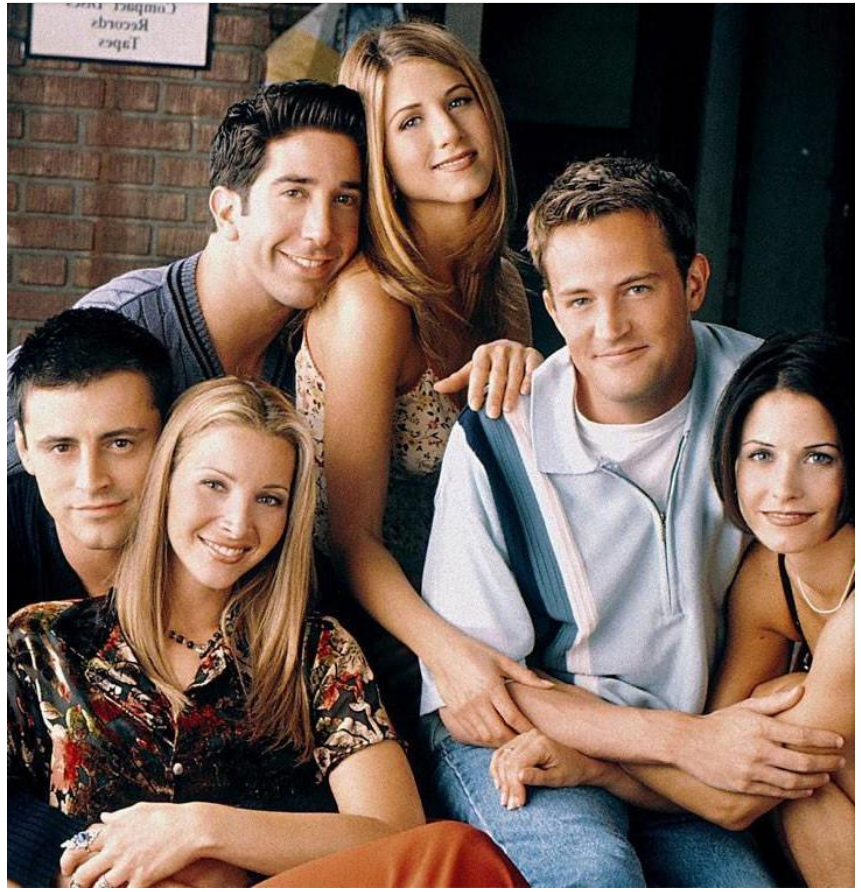
THE ONE ABOUT F•R•I•E•N•D•S

Character Prediction
based on Dialogue


1. Introduction
2. Data Acquisition
3. Data Cleaning
Oh My God!
Seriously?!
PIVOT!!!
4. Machine Learning
5. Results
6. What's next?

THE • ONE • WITH THE • INTRODUCTION



ARE YOU A FRIENDS FANATIC?

The One with all the Lines

- 236 Episodes
- Combination of Beautiful Soup and Regex
- Several episodes were “irregular”
- Used aggregation functions to determine if I needed to re-scrape any data. Total episodes not included = 6, which was only 3% of the total

```
Characters = []
Lines = []
Title = []
Season = []
Episode = []
for url in tqdm(all_episodes_urls):
    r = requests.get(url)
    soup = BeautifulSoup(r.text,'html.parser')
    ps = soup.find_all('p')
    for p in ps:
        char = regex.findall(r"[A-Z][a-zA-Z. ]+:",p.text)
        if char != []:
            if char[0] != "Scene:":
                Characters.append(char[0])
                index = regex.search(char[0], p.text).start() + len(char[0])
                line = p.text[index:]
                Lines.append(line.replace("\n"," "))
                Title.append(soup.title.string)
                season = regex.findall('friends/(\w+\d+)', url)
                Season.append(season)
                ep = regex.findall('friends/\w+\d+/(\d+)', url)
                Episode.append(ep)
```

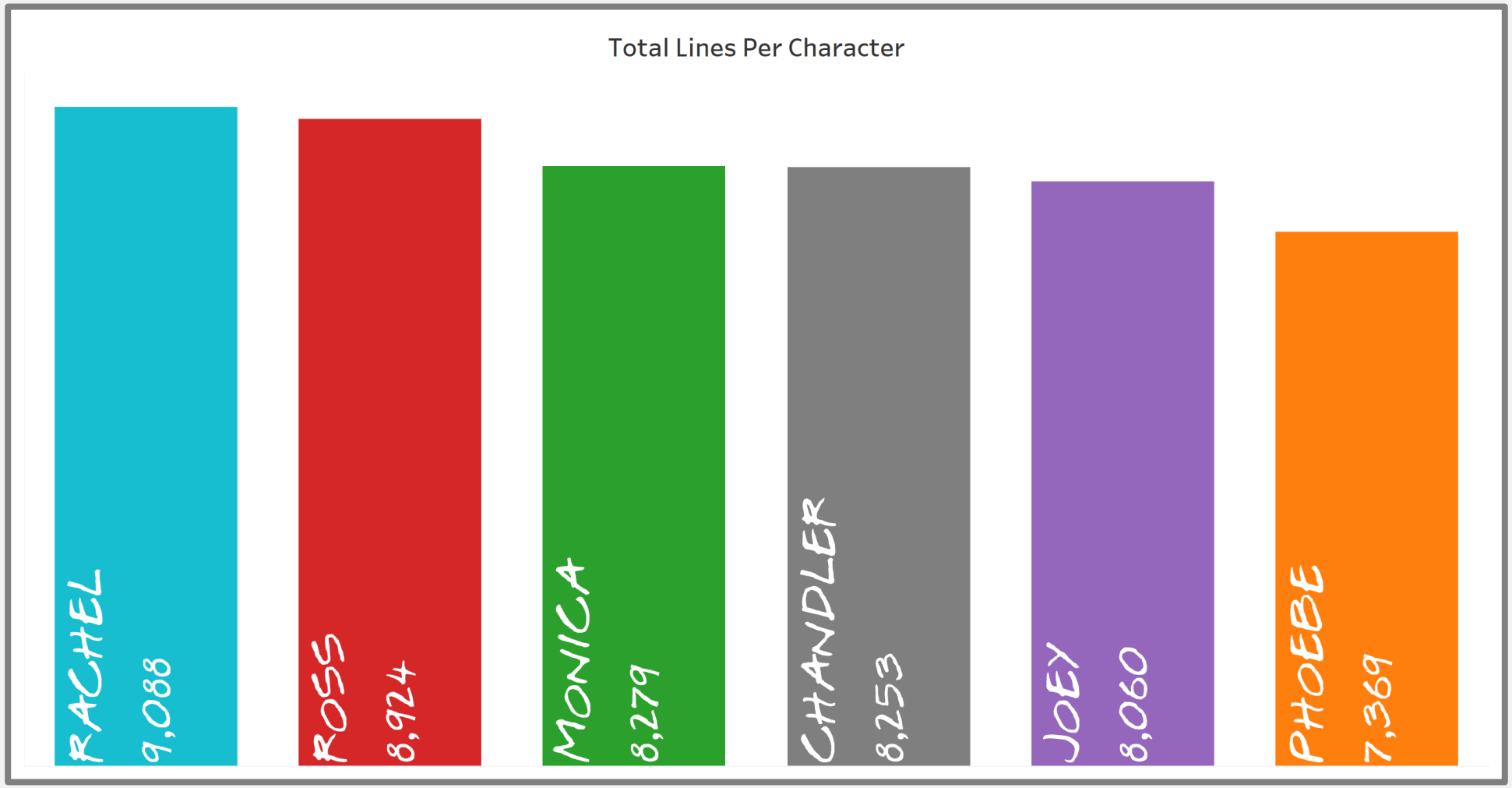
The • One • with • all • the • Cleaning

- Reviewed the data using methods noted below
- Created a Name function to correct misspellings
- Dropped all null lines and lines from "Other" cast members
- Used a series of regex.sub(pattern, '', line) to replace non-dialogue

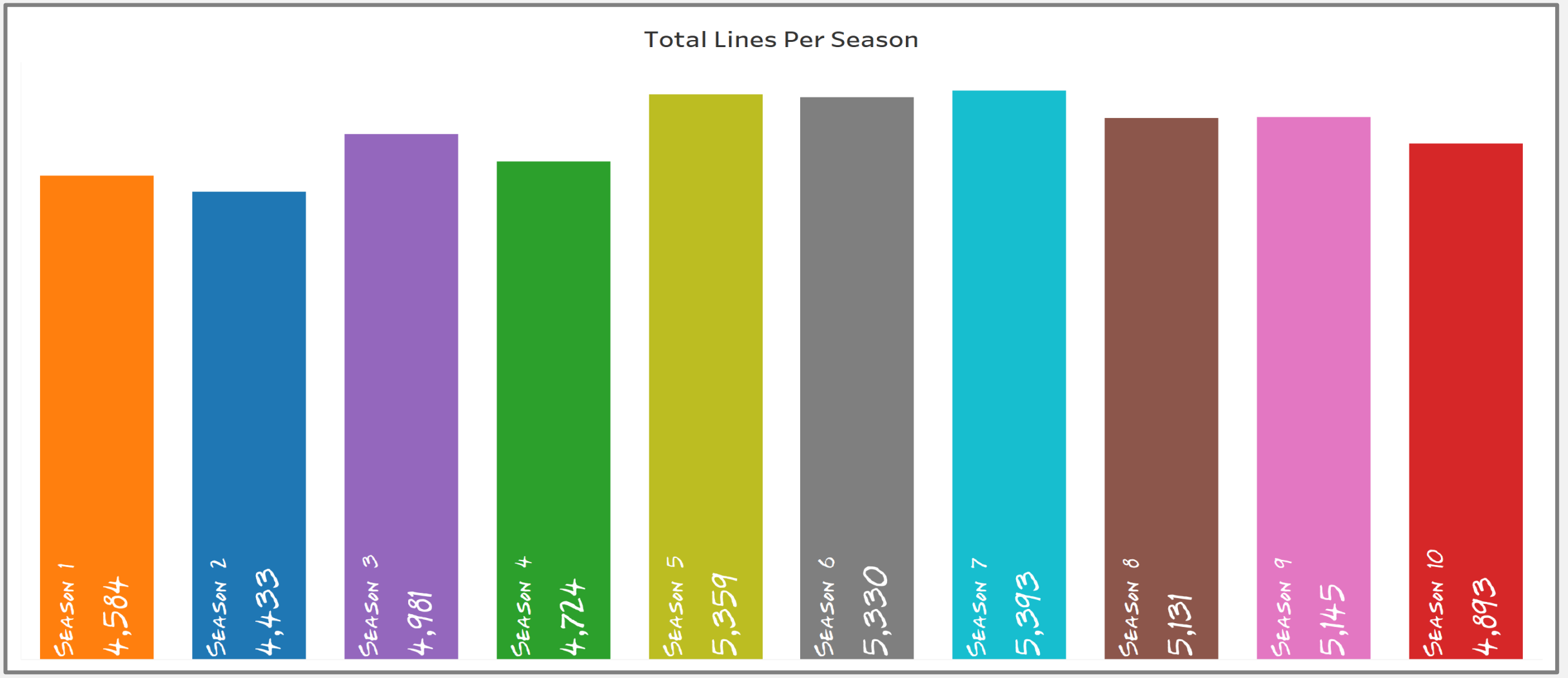
```
Ok. (She sits on the bed and Ross sits near her)<+i>
 Thank you for coming with me today.
(to Ross) Let me get you some coffee.
About 20 minutes. <font size="4">  <b> CLOSING CREDI
TS </b> </font>
Well, I tracked down Marcel and get this, he's healt
hy, he's happy, and he's right there in New York fil
ming <i> Outbreak II - The Virus Takes Manhattan </i
> .
```

- `df.info()`
- `df.isnull().sum()`
- `df.dropna()`
- `df.Character.isin(main_characters).sum()`
- `set(df.Character)`
- `def name_func(char)`
- `df.groupby('Character').count().loc[main_characters]`

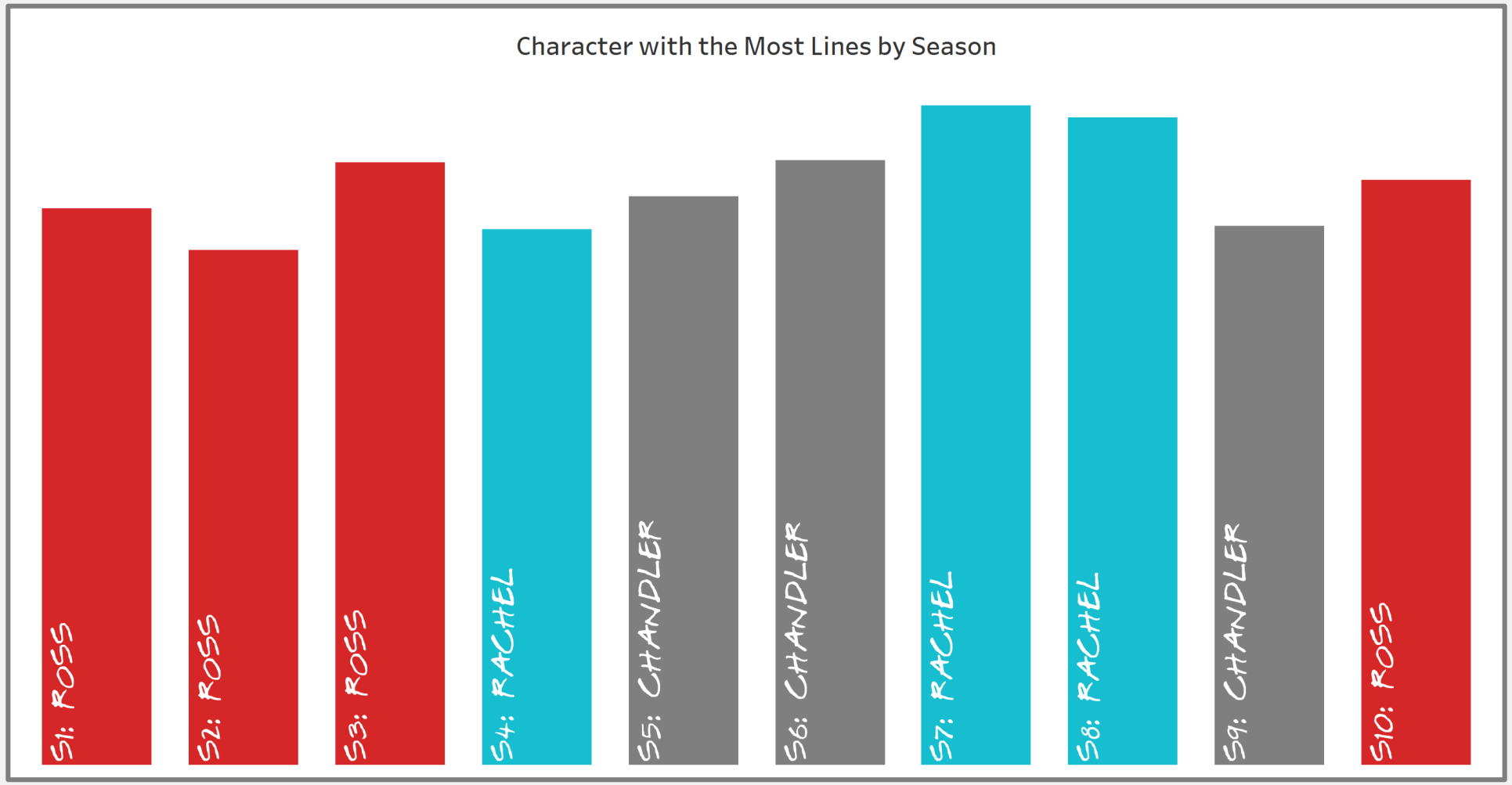
The One with the Exploration Mission



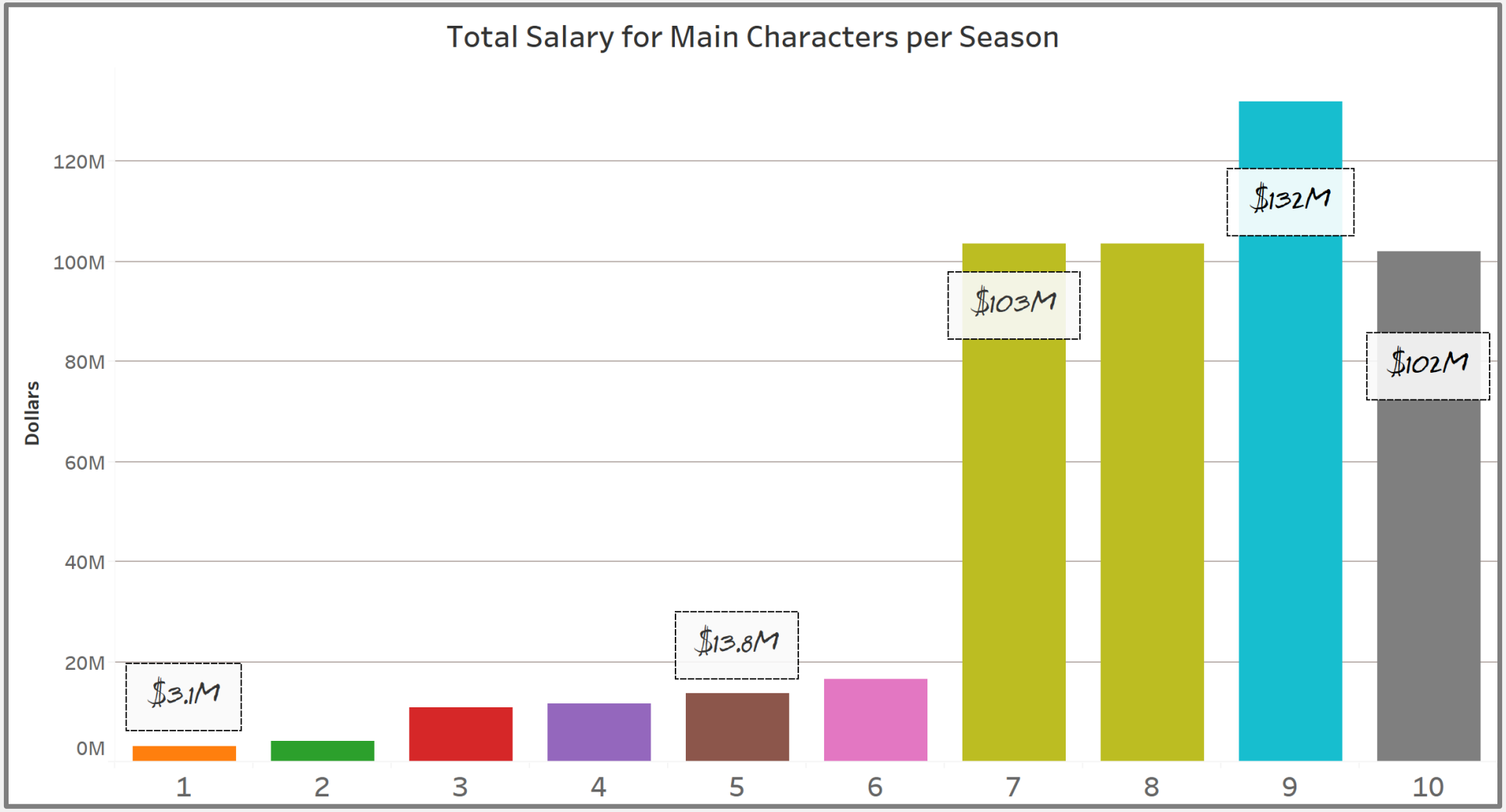
The • One • with • the • Exploration • Mission



The • One • with • the • Exploration • Mission



THE • ONE • WITH • THE • EXPLORATION • MISSION



The One with the Models

	Character	Baseline Percent
0	Rachel	18.2%
1	Ross	17.9%
2	Monica	16.6%
3	Chandler	16.5%
4	Joey	16.1%
5	Phoebe	14.7%
6	Average	16.7%

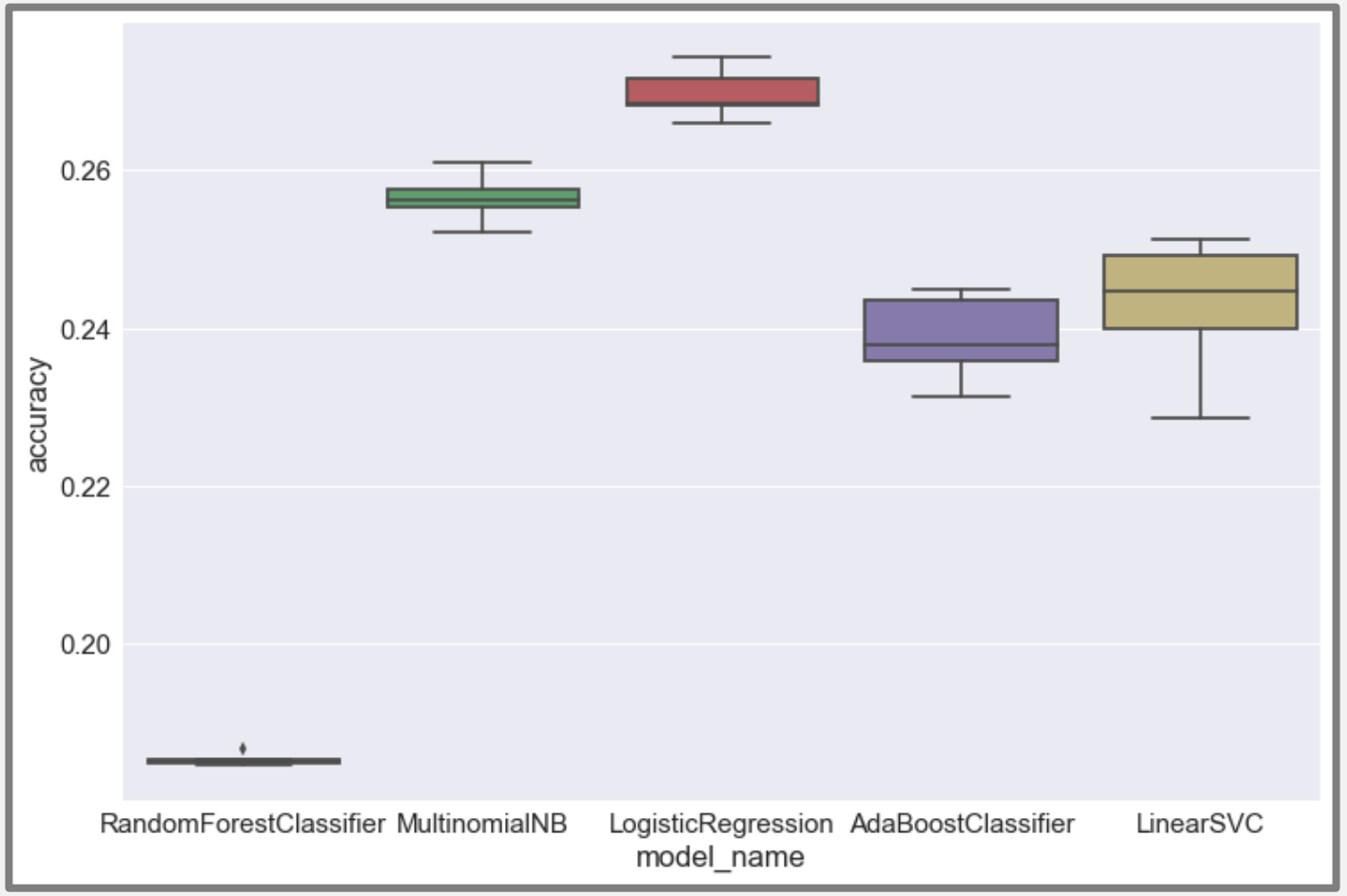
Models Used

- Logistic Regression
- Decision Tree Classifier
- Linear SVC
- Random Forest Classifier
- Extra Trees Classifier
- AdaBoost
- Multinomial Naive Bayes
- SGD Classifier

The • One • with • the • Models

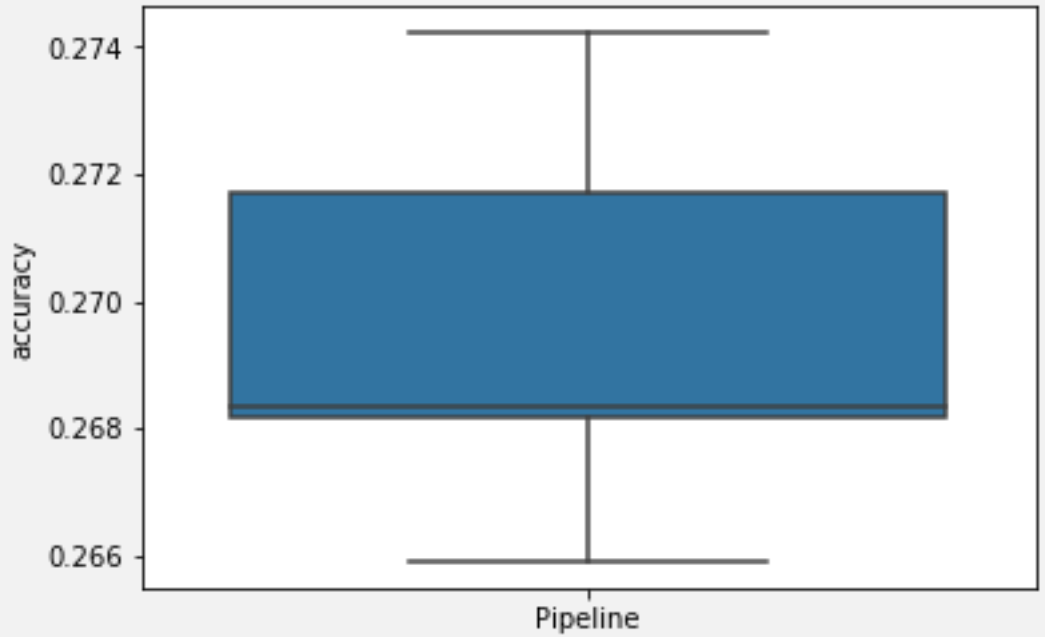
What I learned:

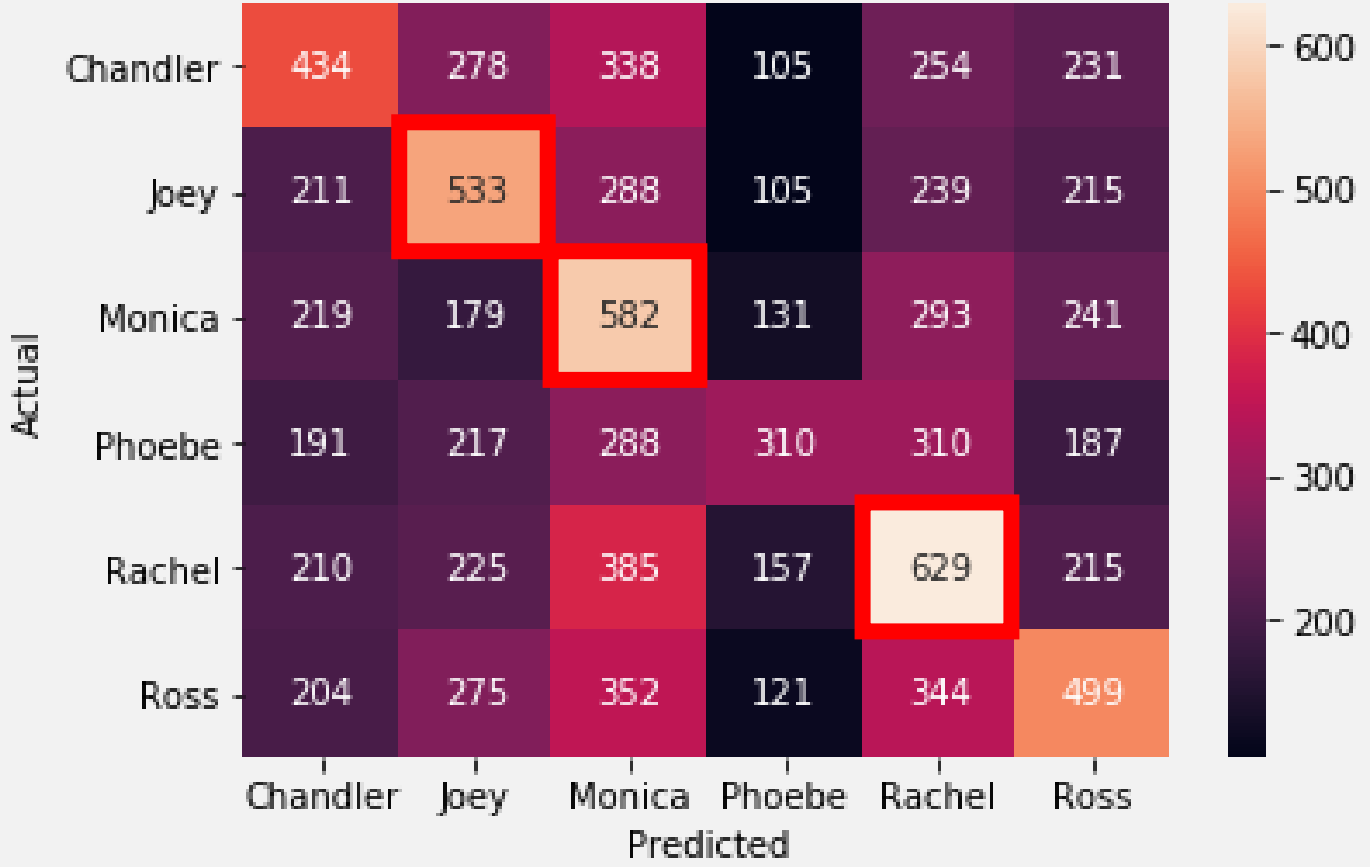
1. Gridsearch is **slow**
2. Pipelines are more efficient
3. PCA – lose information, didn't use to test predictions
4. Know when to **STOP**



The • One • with • the • Results

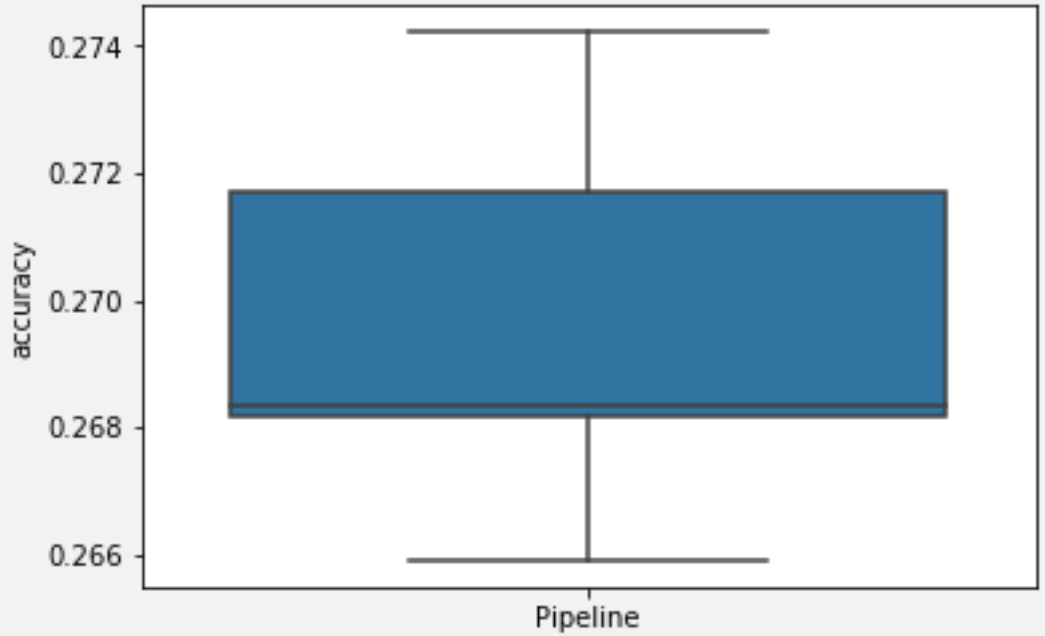
- The Winner is…. Logistic Regression!
  - Preprocessor: Count Vectorizer
  - Ngram Range: 1-3
  - Stop Words = English
  - Accuracy Score: 0.2988
  - Number of features: 219,010

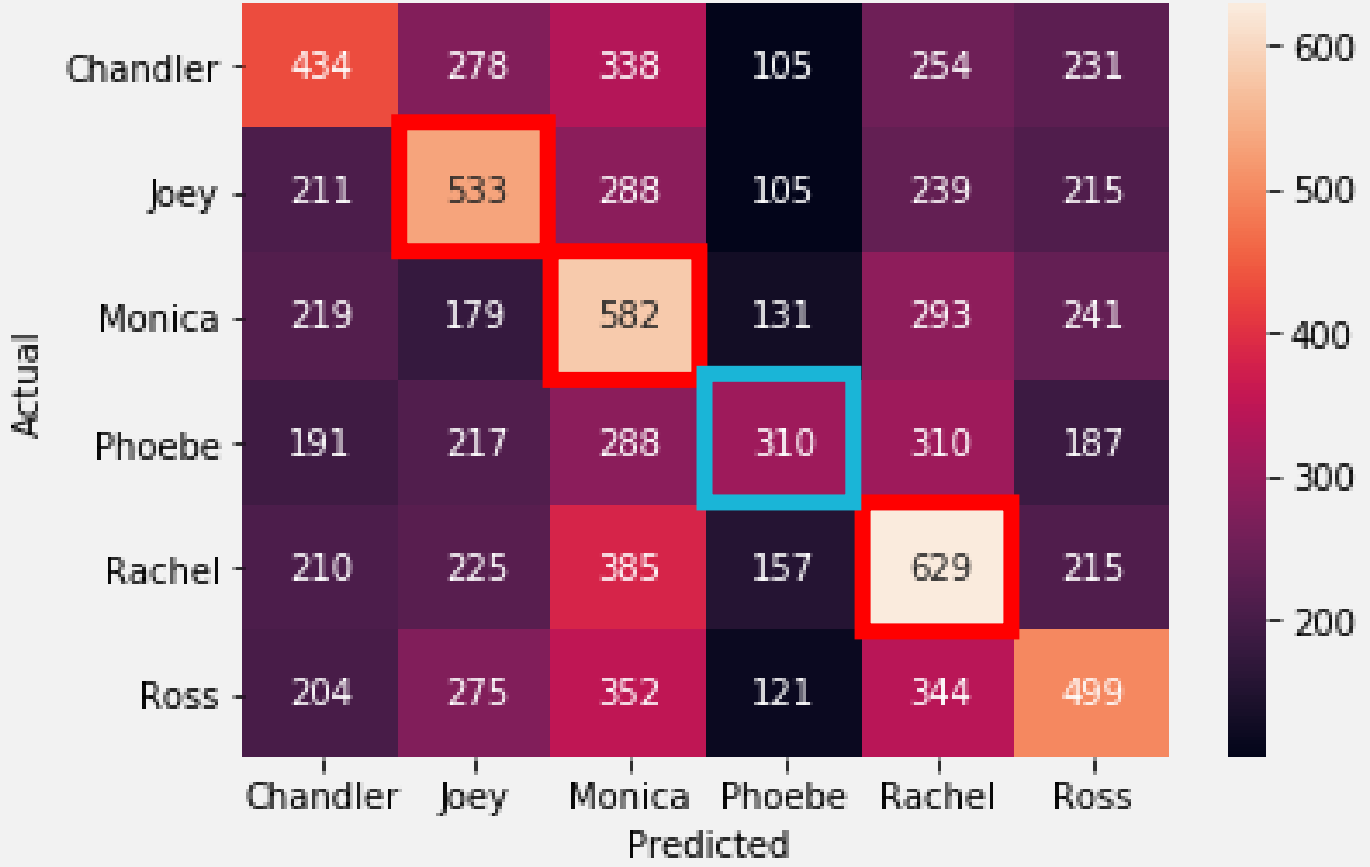




The • One • with • the • Results

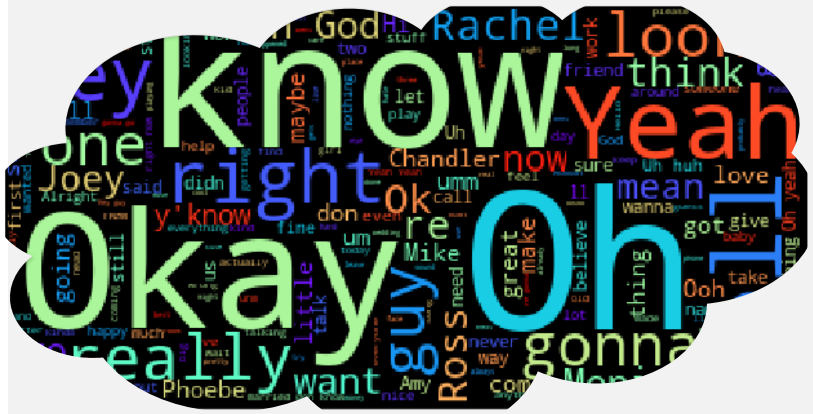
- The Winner is…. Logistic Regression!
  - Preprocessor: Count Vectorizer
  - Ngram Range: 1-3
  - Stop Words = English
  - Accuracy Score: 0.2988
  - Number of features: 219,010

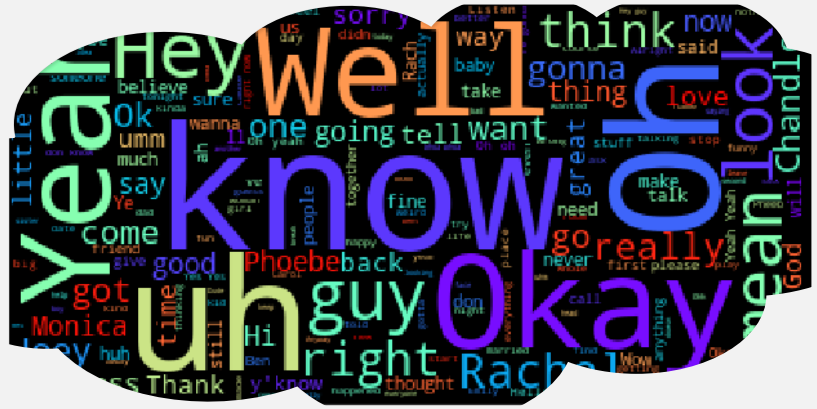


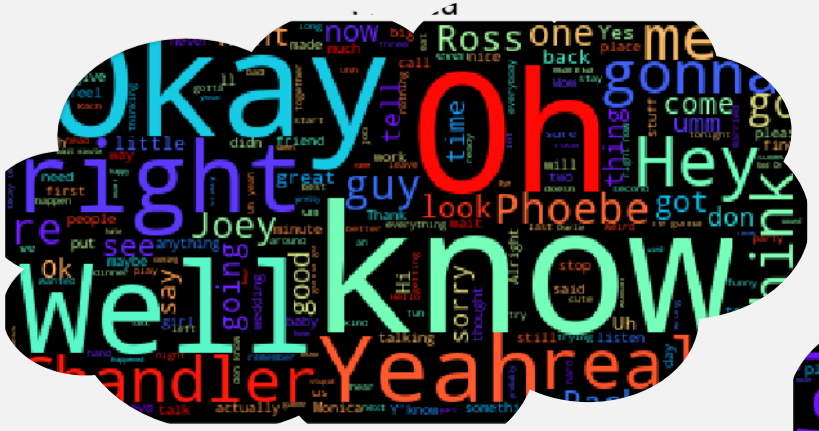


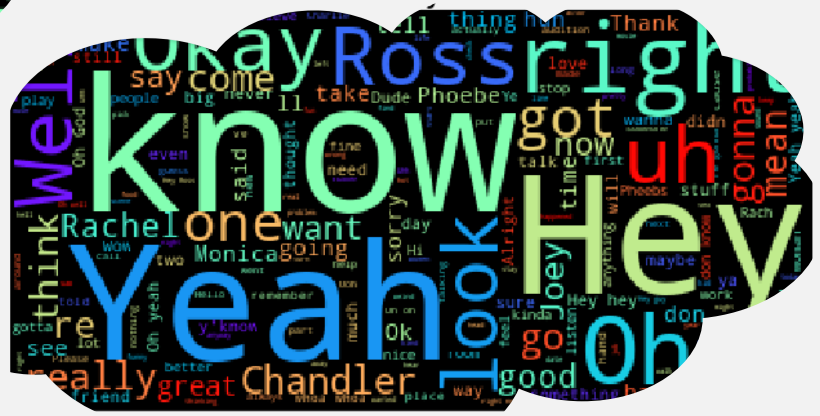
Word Clouds

Can you tell who is who?



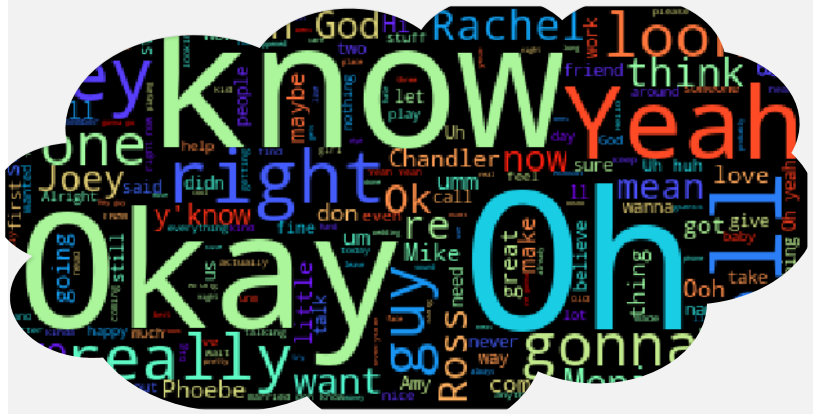






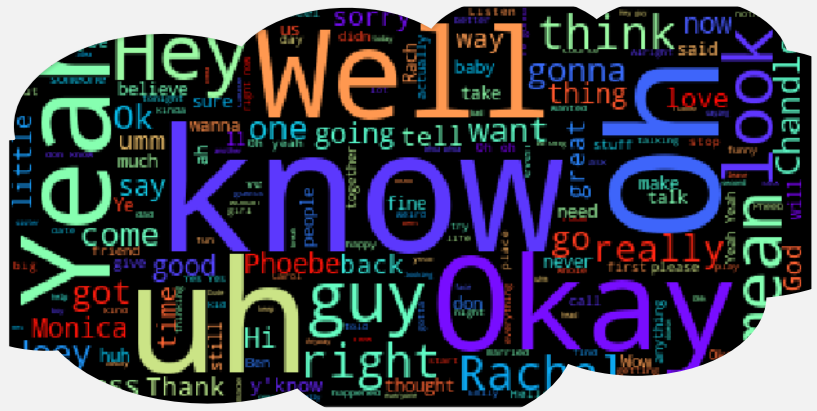
Word Clouds

Phoebe

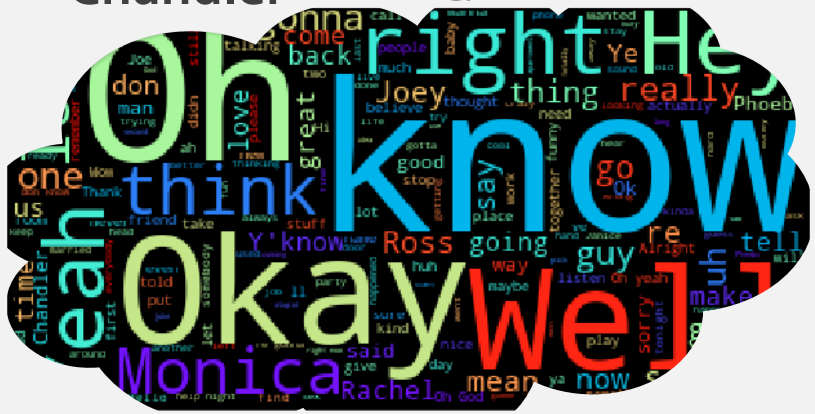


Rachel

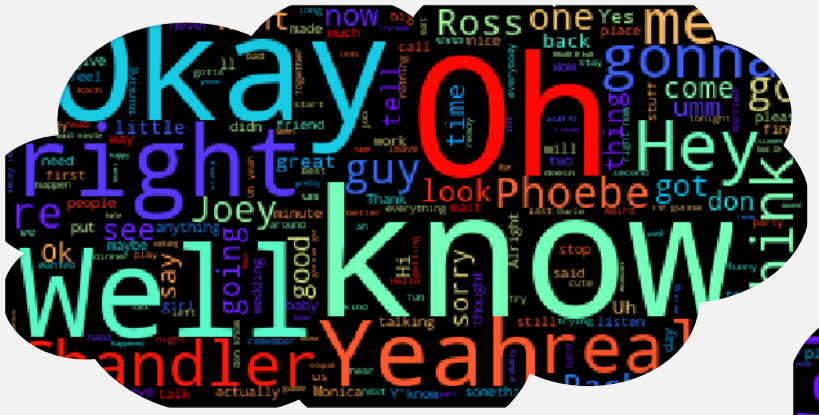
Ross



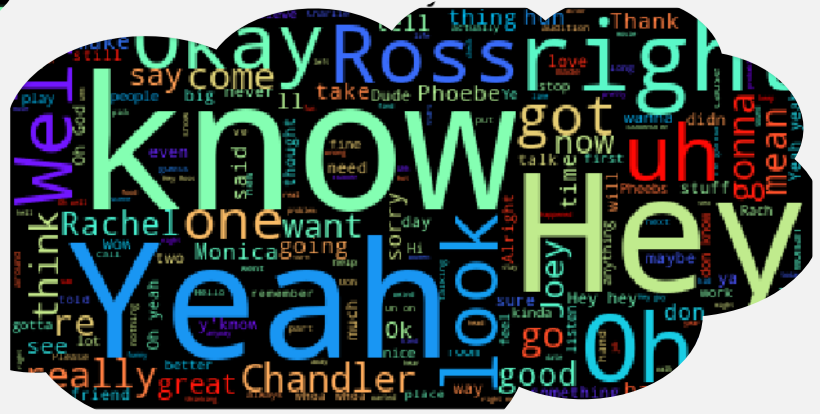
Chandler



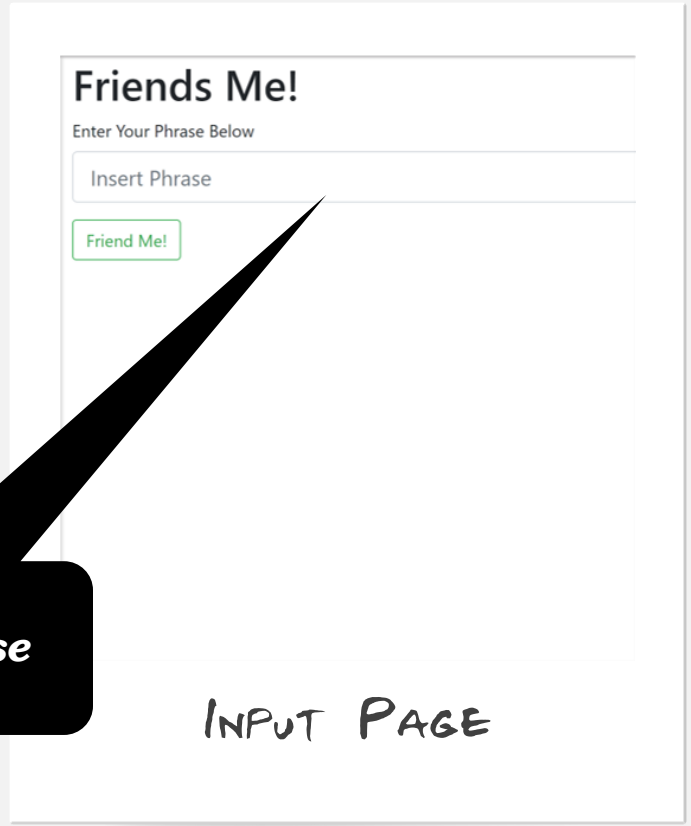
Monica

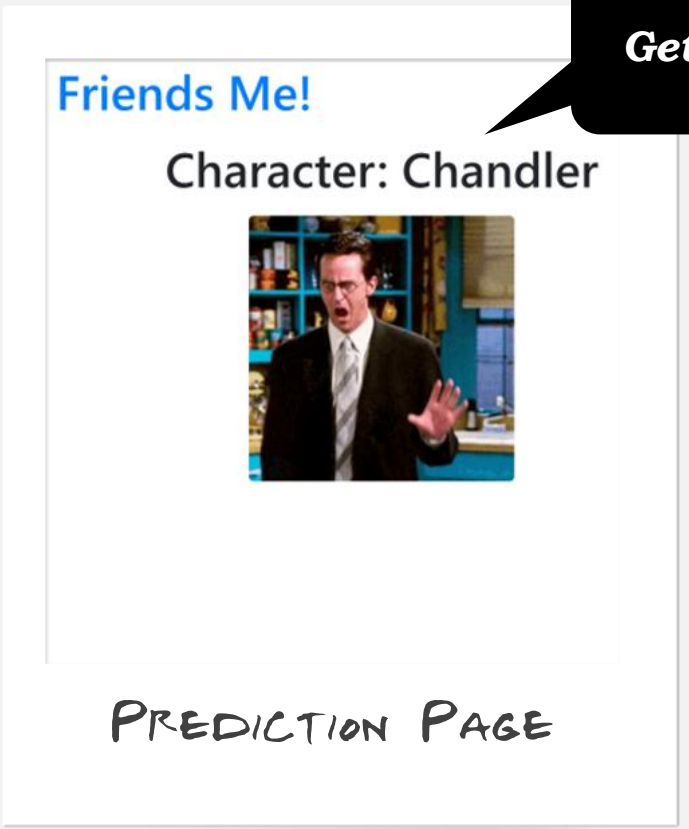


Joey



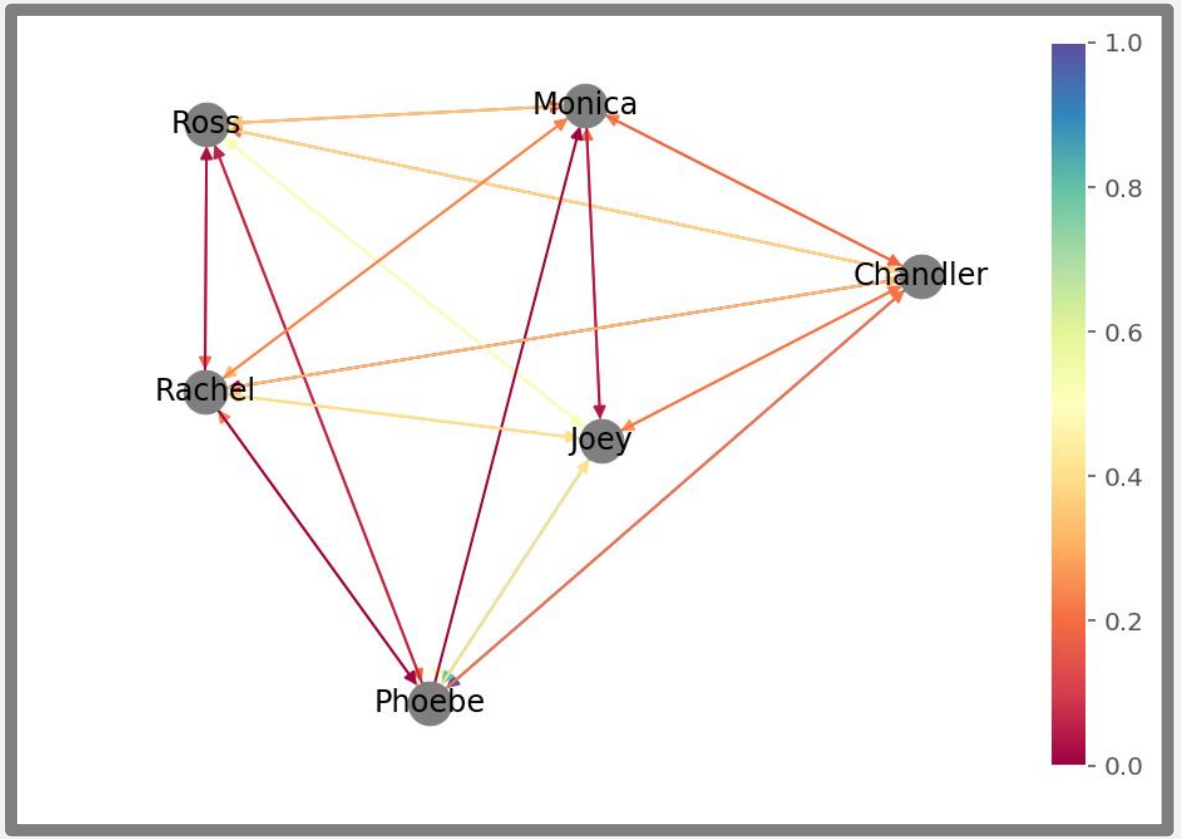
Flask • on • Fleek





The One with the Future...Steps

- What else can we do?
  - Download someone's twitter feed to predict which Friends character they are most like
  - Determine the centrality of the characters



Thanks!