Mystery Machine Learning

and the Ghosts of Text Data

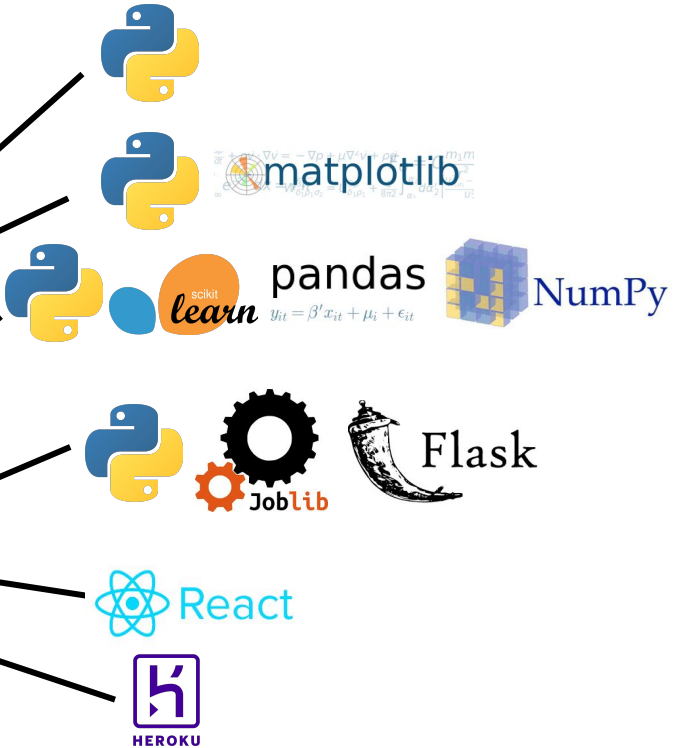
DataRobot

DATA SCIENCE

1. Define problem
2. Create data set
3. Explore data
4. Train your model
5. Evaluate your model
6. Deploy your model
7. Build your app
8. Deploy your app

DATA SCIENCE

1. Define problem
2. Create data set
3. Explore data
4. Train your model
5. Evaluate your model
6. Deploy your model
7. Build your app
8. Deploy your app



How to draw an owl

1.

2.

1. Draw some circles

2. Draw the rest of the [redacted]ing owl

DataRobot

1: Define the Problem

How might you use machine learning at your company?

DataRobot

Goal: Predict which Scooby Doo character said the line

DataRobot

DataRobot

DataRobot

Play this video...

https://www.youtube.com/watch?v=DoJVJpTruzg

DataRobot

Goal: Predict which Scooby Doo character said the line

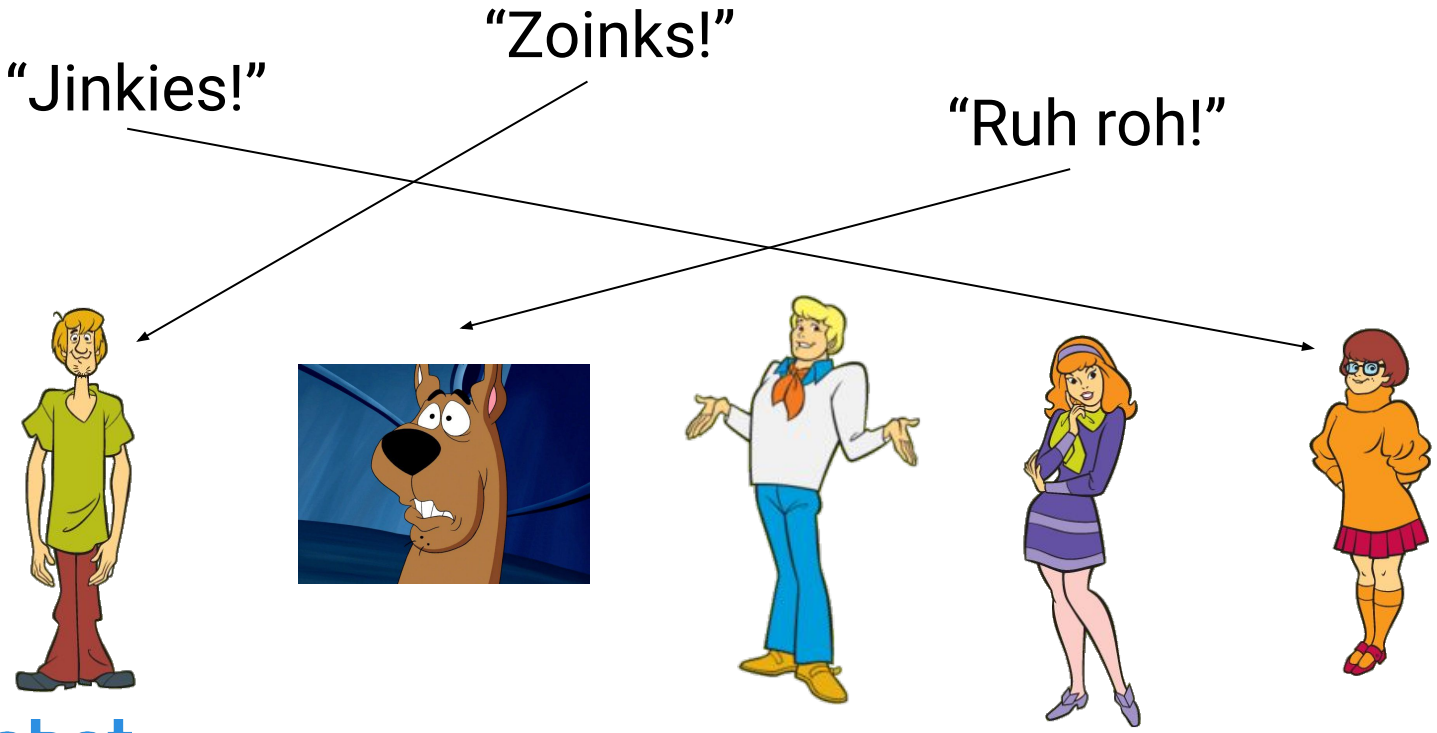
“Jinkies!”

“Zoinks!”

“Ruh roh!”

DataRobot

Goal: Predict which Scooby Doo character said the line



DataRobot

2: Create the Dataset

Step 2:
Create Dataset

Transcripts Wiki

POPULAR PAGES COMMUNITY EXPLORE

in: *Scooby-Doo Transcripts, Movies*

Scooby-Doo! and the Monster of Mexico

EDIT

Contents [show]

Scene 1

(*We see a lake near a city called Veracruz, there was the Plaza where there was people and a mirachi band and we see a man playing the Marimba and his son a toy marimba and with his chihuahua dancing and heard a sound from the alley and chases it*)

Jorge Otero: Chiquita! (*He run towards her and follows her, later a man is still playing but noticed his son is missing*)

Alejo Otero: Jorge?! Jorge! Jorgito! (He runs off to find Jorge)

(*Later, Chiquita follows the shadow through the alley and catches her breath at the pier she looks around and sees a light green glow and sees a monster while Jorge picks her up and sees the monster, they were frightened*

https://transcripts.fandom.com

Step 2:
Create Dataset

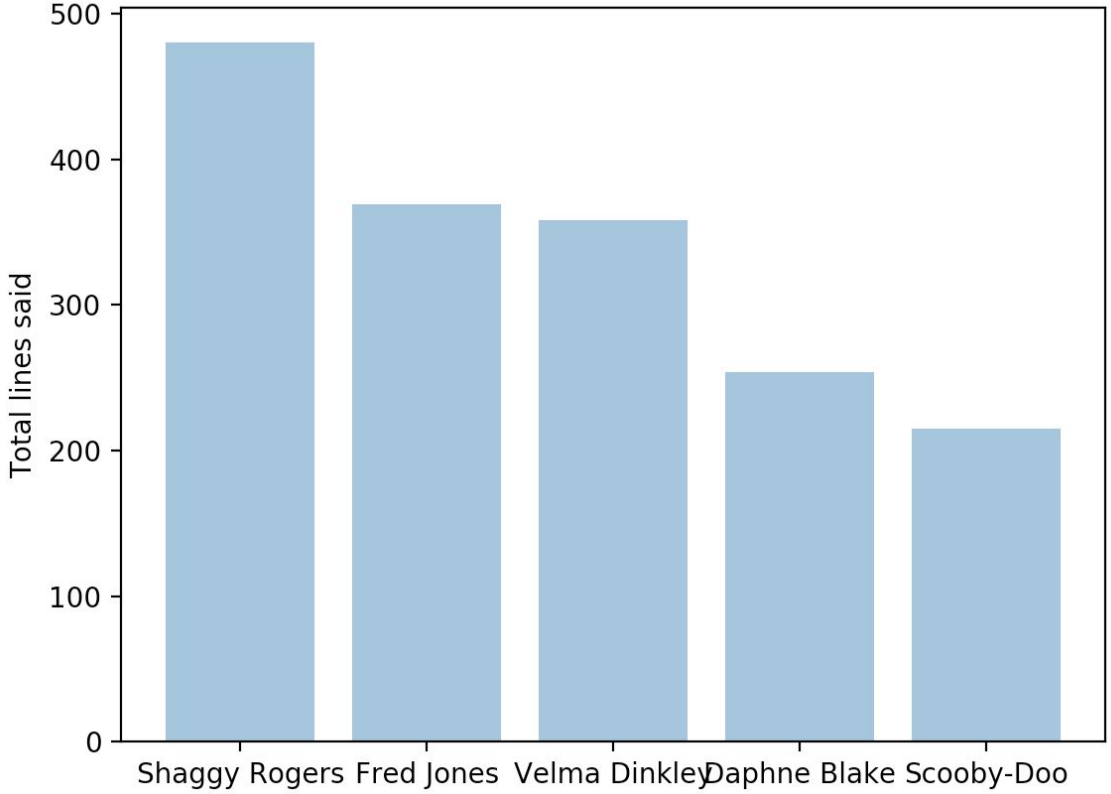
```
transcripts = os.listdir('transcripts')
all_lines = []
for transcript in transcripts:
    with open('transcripts/{}'.format(transcript), 'r') as transcript_file:
        lines = transcript_file.read()
    lines = lines.split('\n')
    all_lines.append(lines)
all_lines = sum(all_lines, [])
```

Step 2:
Create Dataset

```
for line in all_lines:
    for character in CHARACTERS:
        alias = get_first_name(character)
        character_string = '{}: '.format(character)
        alias_string = '{}: '.format(alias)
        line = remove_parentheticals(line)
        if character_string in line or alias_string in line:
            statement = line.replace(character_string, '')
            statement = statement.replace(alias_string, '')
            statement = clean_punct(statement)
            statements[character].append(statement)
```

3: Explore Data

Step 3:
Explore Data

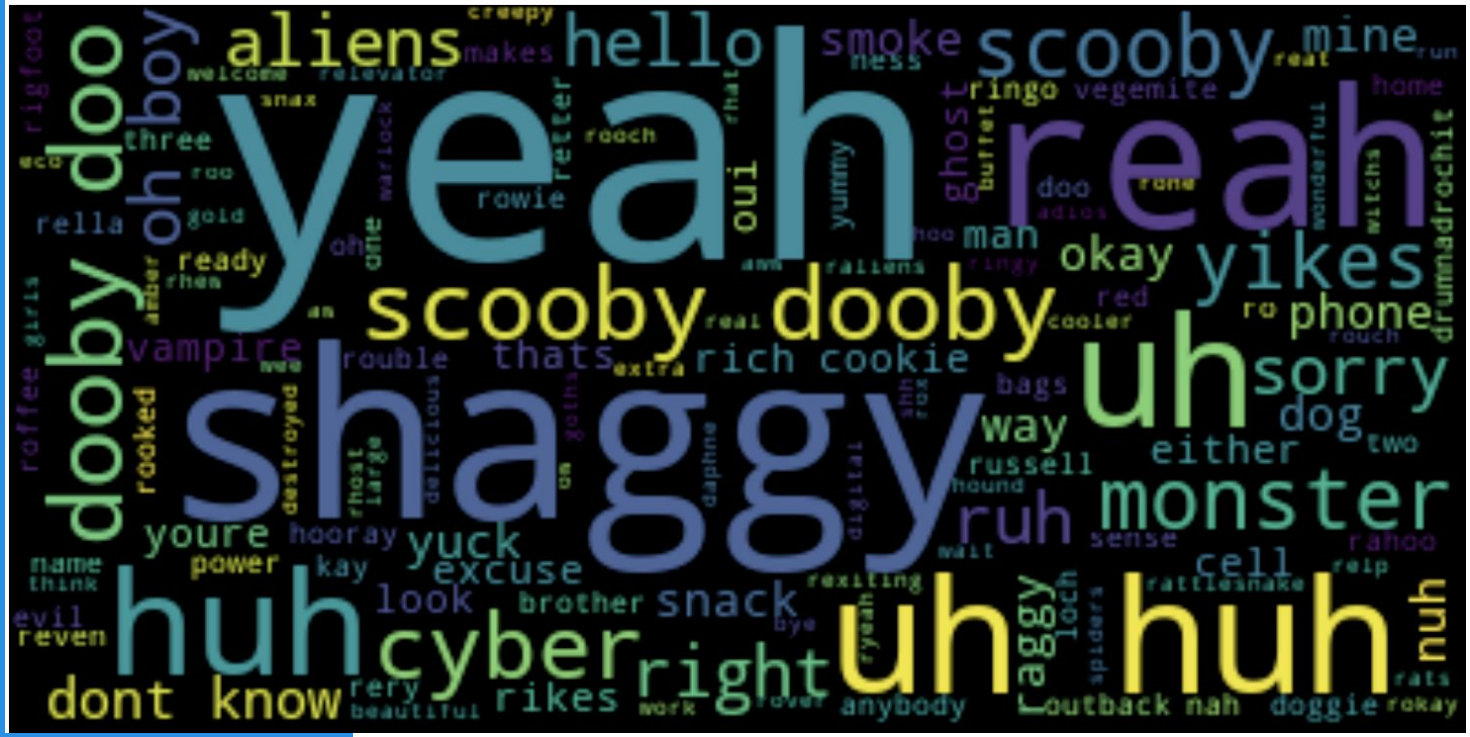


Step 3:
Explore Data

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

lines = pd.read_csv('scooby_doo_lines.csv')

character_counts = lines['character'].value_counts()
y_pos = np.arange(len(character_counts))

labels = character_counts.keys()
y_pos = np.arange(len(labels))
counts = character_counts.values

plt.bar(y_pos, counts, align='center', alpha=0.5)
plt.xticks(y_pos, labels)
plt.ylabel('Total lines said')

plt.show()
```

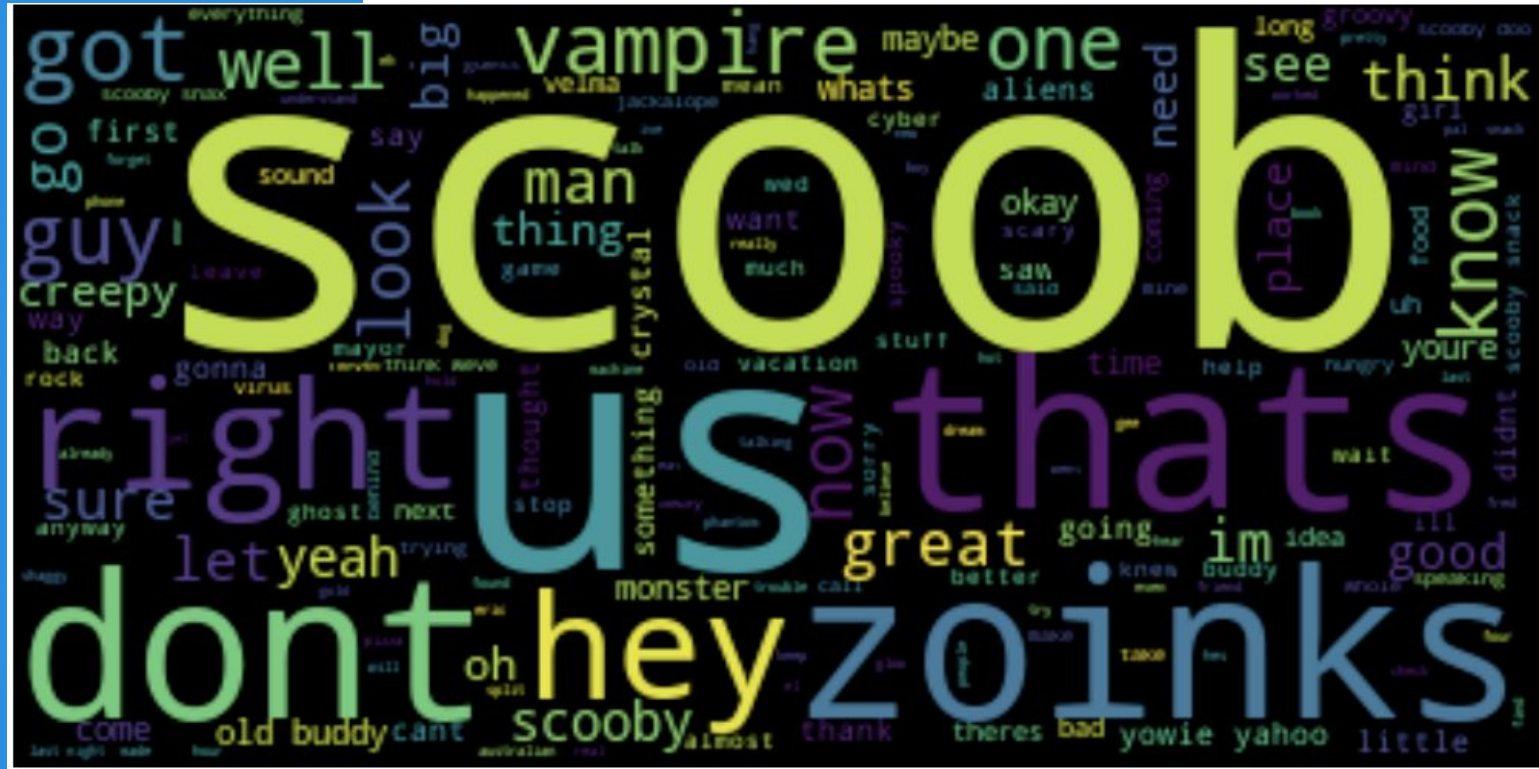
Step 3:
Explore Data

Scooby says...



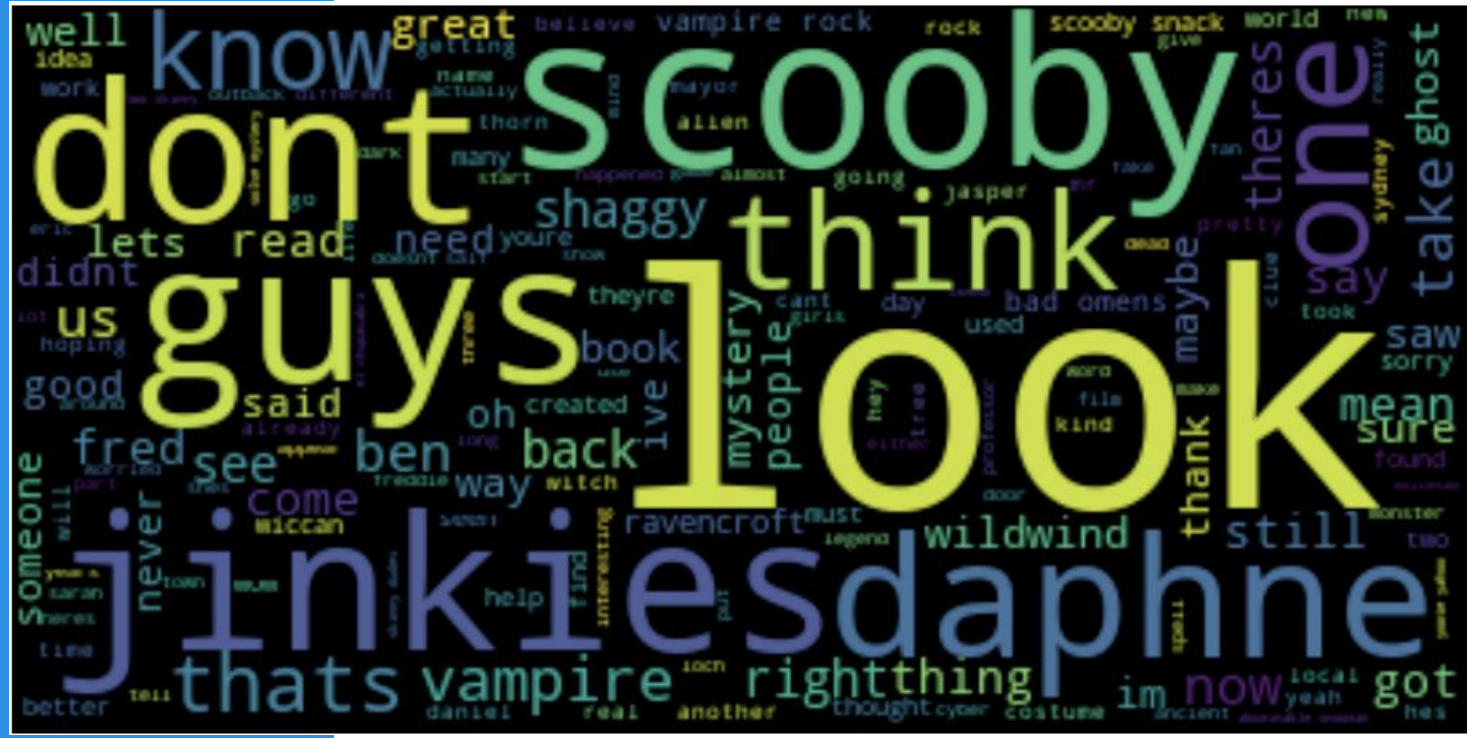
Step 3:
Explore Data

Shaggy says...



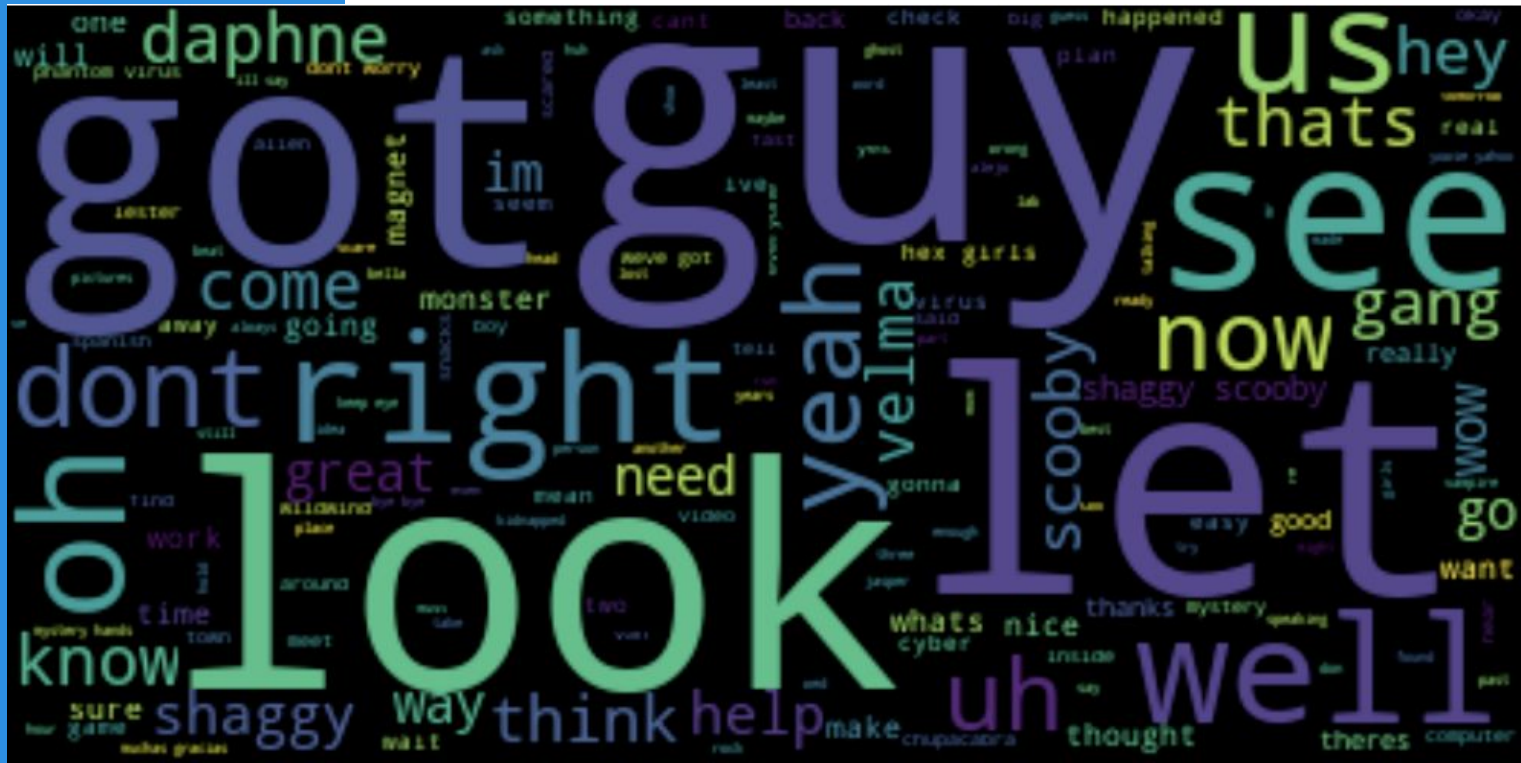
Step 3:
Explore Data

Velma says...



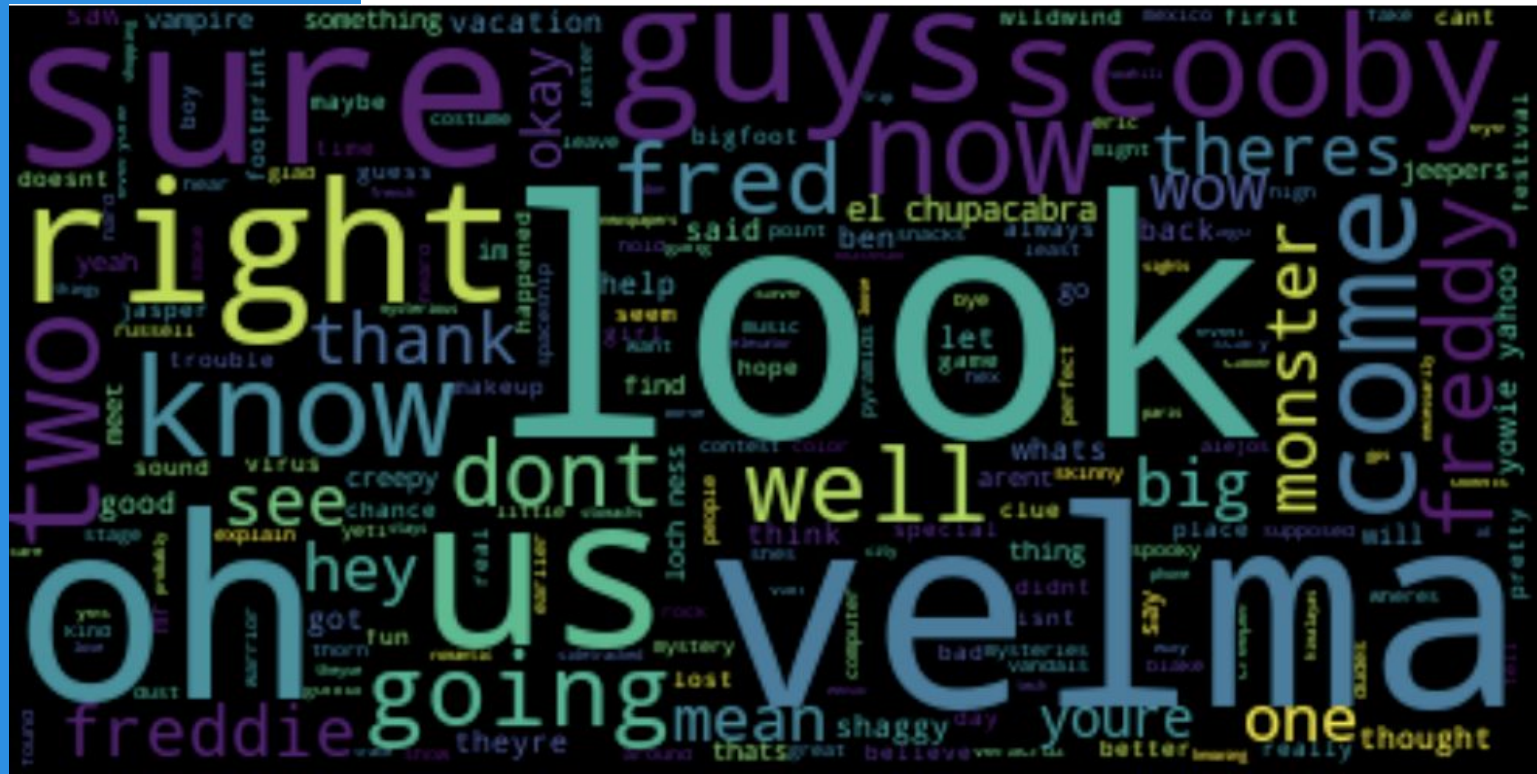
Step 3:
Explore Data

Fred says...



Step 3:
Explore Data

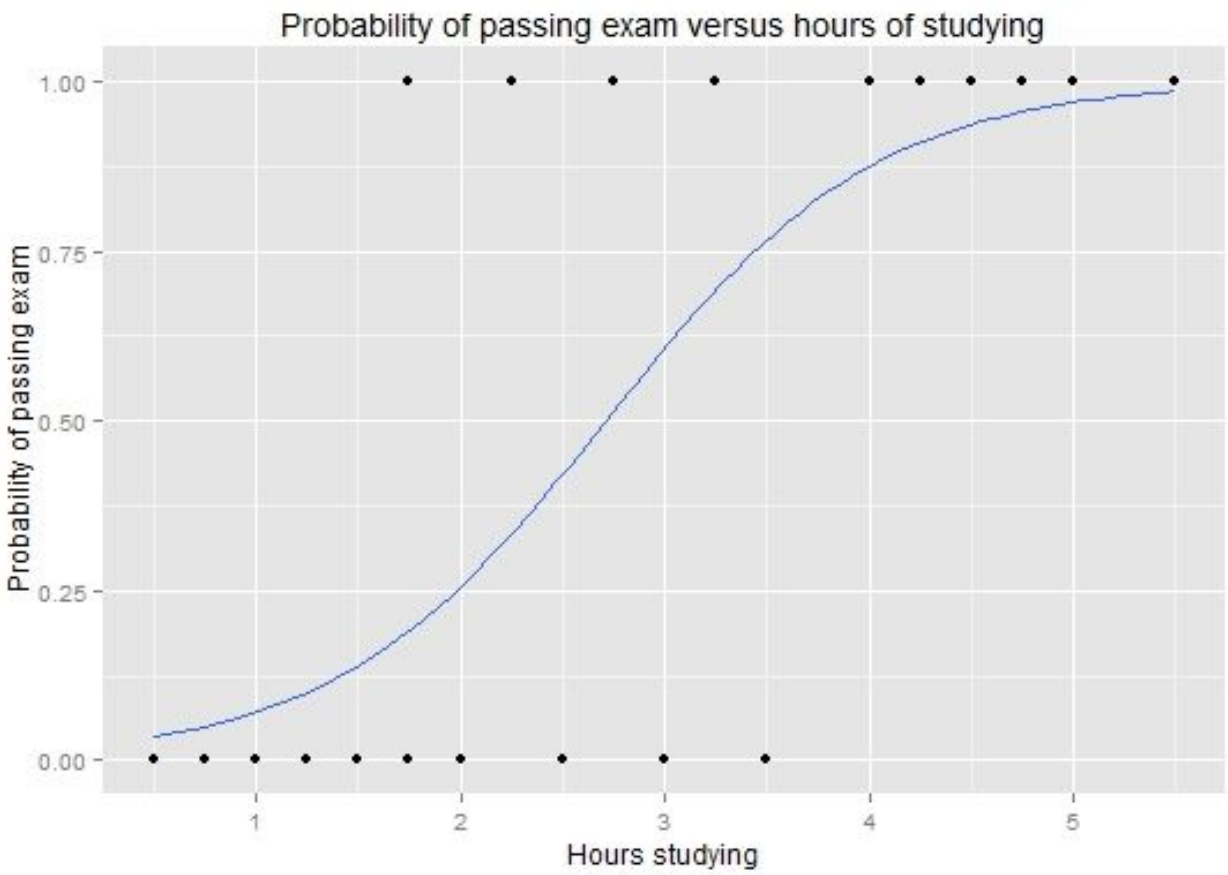
Daphne says...



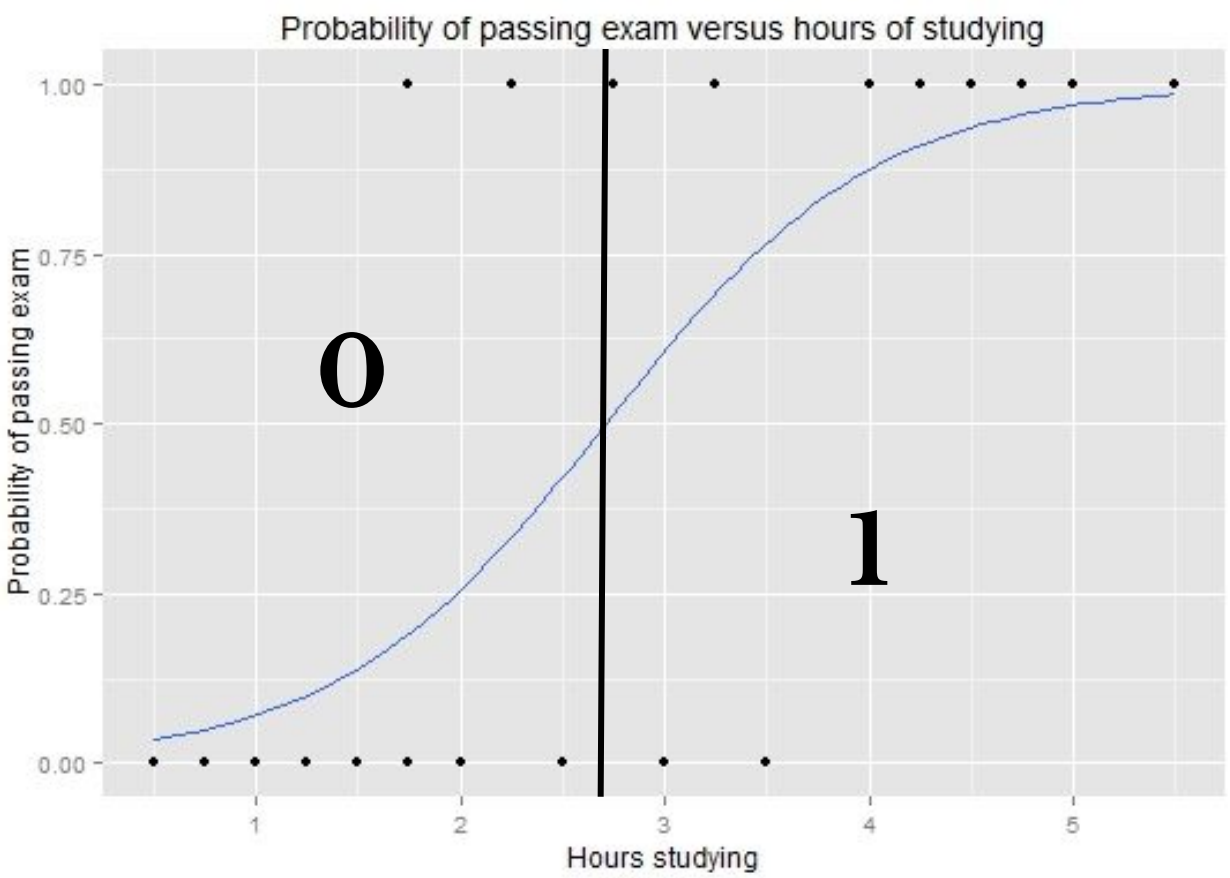
```python
import pandas as pd
import matplotlib.pyplot as plt

from wordcloud import WordCloud


def plot_character_words(character):
    lines = pd.read_csv('scooby_doo_lines.csv')
    lines = lines[lines['character'] == character]  # Filter to character
    lines = lines['line']                           # Get the text data
    lines = [l.split(' ') for l in lines]           # Split each line into individual words
    lines = sum(lines, [])                          # Turn into single list of words
    lines = ' '.join(lines)                         # Convert into string for wordcloud
    wordcloud = WordCloud().generate(lines)         # Make word cloud
    plt.imshow(wordcloud, interpolation='bilinear') # Plot
    plt.axis('off')
    plt.show()

plot_character_words('Scooby-Doo')
```

4: Train Your Model

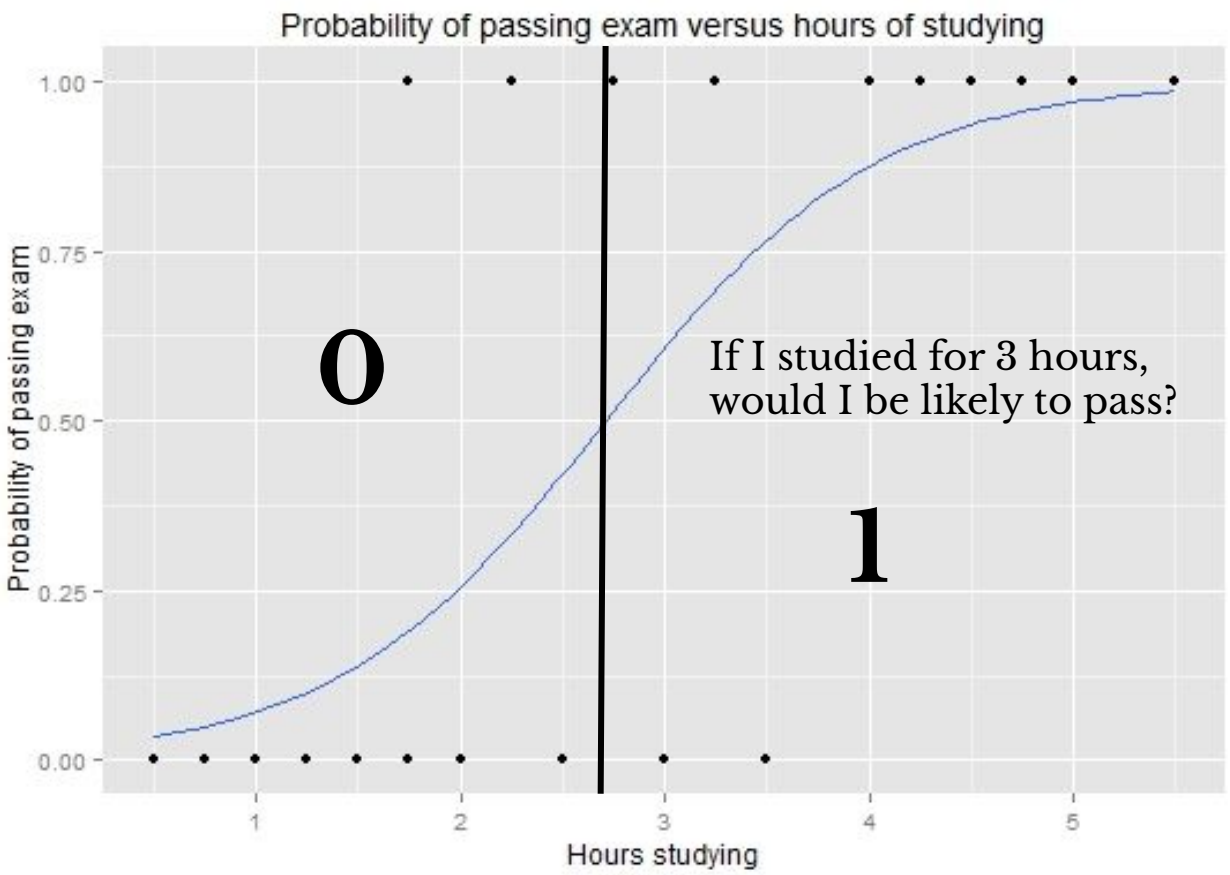
Step 4:
Train Your Model



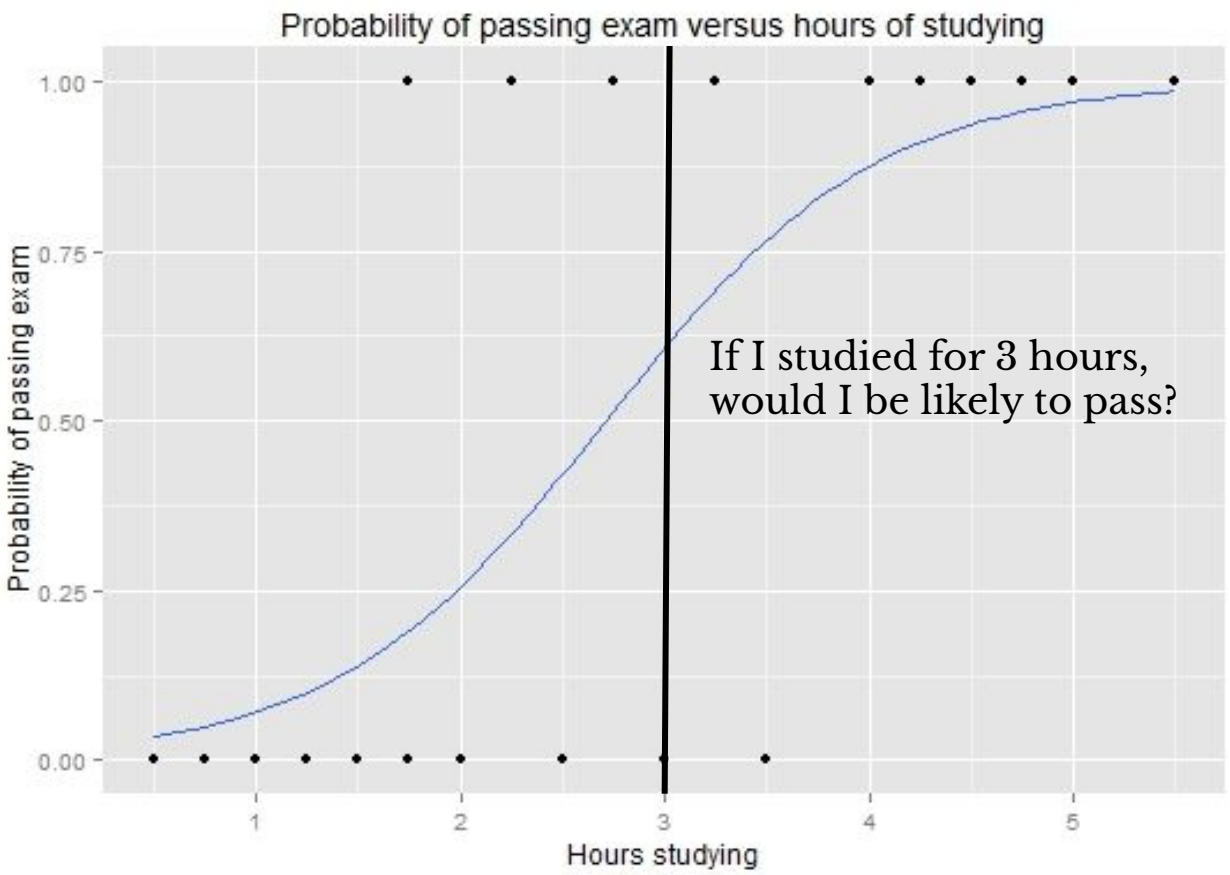
Step 4:
Train Your Model



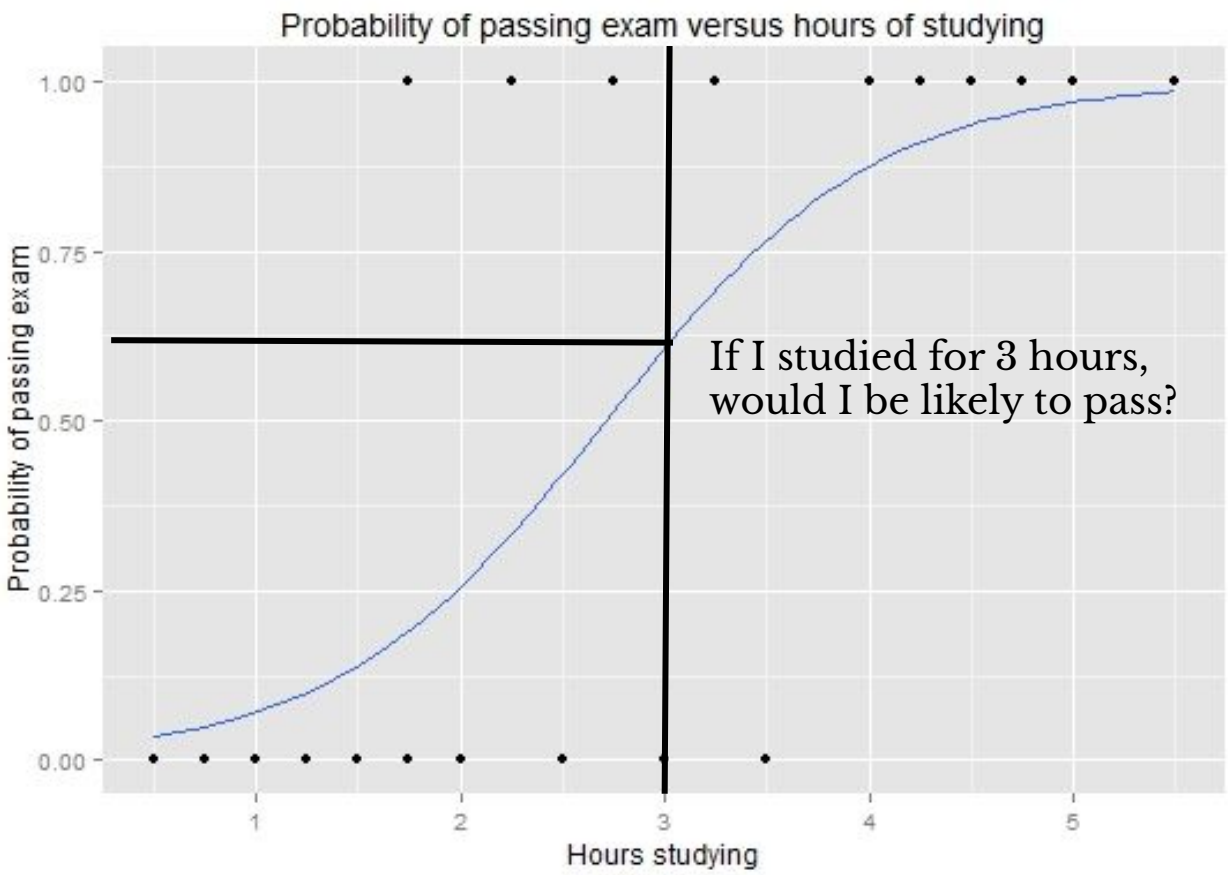
Step 4:
Train Your Model



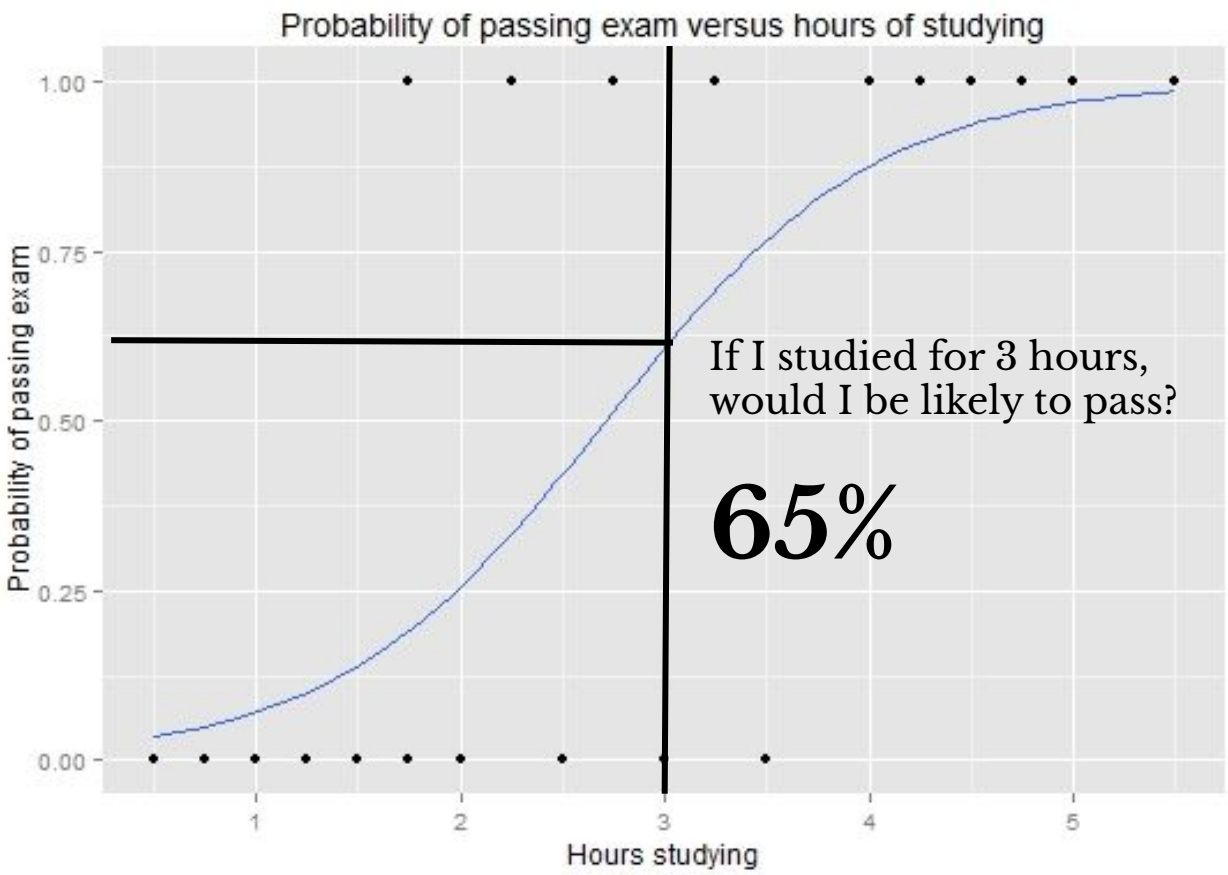
Step 4:
Train Your Model



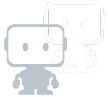
Step 4:
Train Your Model



Step 4:
Train Your Model



Step 4:
Train Your Model



Sklearn Models

1. **Fit model**

```
from sklearn.linear_model import
LogisticRegression
model = LogisticRegression()
model.fit(data, target)
```

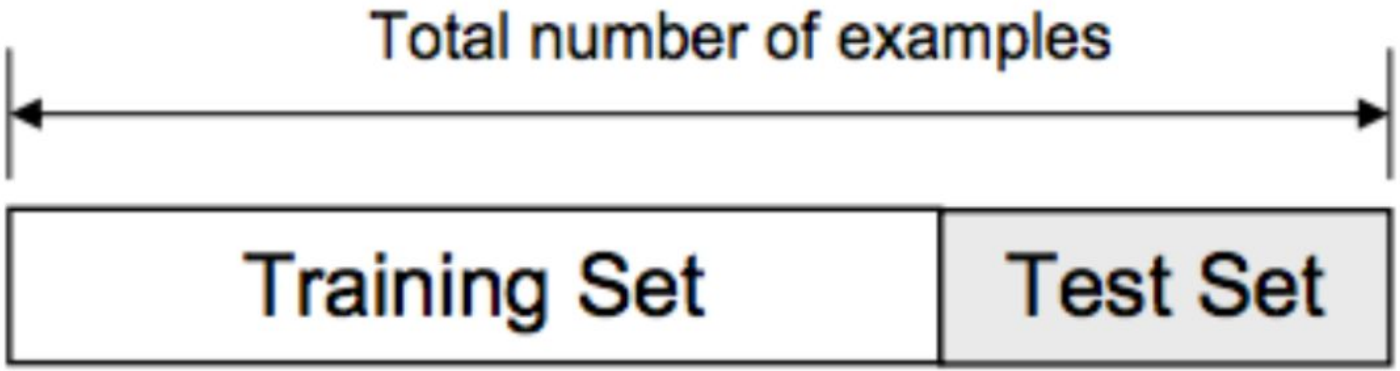
2. **Predict**

```
model.predict(new_data)
```

Step 4:
Train Your Model

Train-Test Split

Make sure your model works on data it hasn't seen before.



Step 4:
Train Your Model

Multiclass

Step 4:
Train Your Model

Multiclass

To predict **Shaggy vs. Scooby vs. Fred vs. Daphne vs. Velma**…

Step 4:
Train Your Model

Multiclass

To predict **Shaggy vs. Scooby vs. Fred vs. Daphne vs. Velma**…

1.) Predict **Shaggy vs. Not Shaggy**
2.) Predict **Scooby vs. Not Scooby**
3.) Predict **Fred vs. Not Fred**
4.) Predict **Daphne vs. Not Daphne**
5.) Predict **Velma vs. Not Velma**
6.) Combine

Step 4:
Train Your Model

Text Data

Step 4:
Train Your Model

Text Data

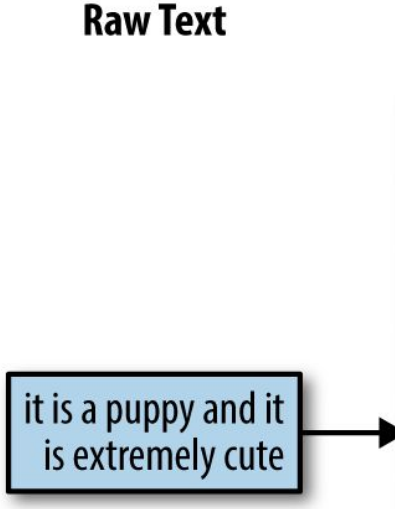
Problem: Models work on a vector of numbers, not a list of words.

Step 4:
Train Your Model

Text Data

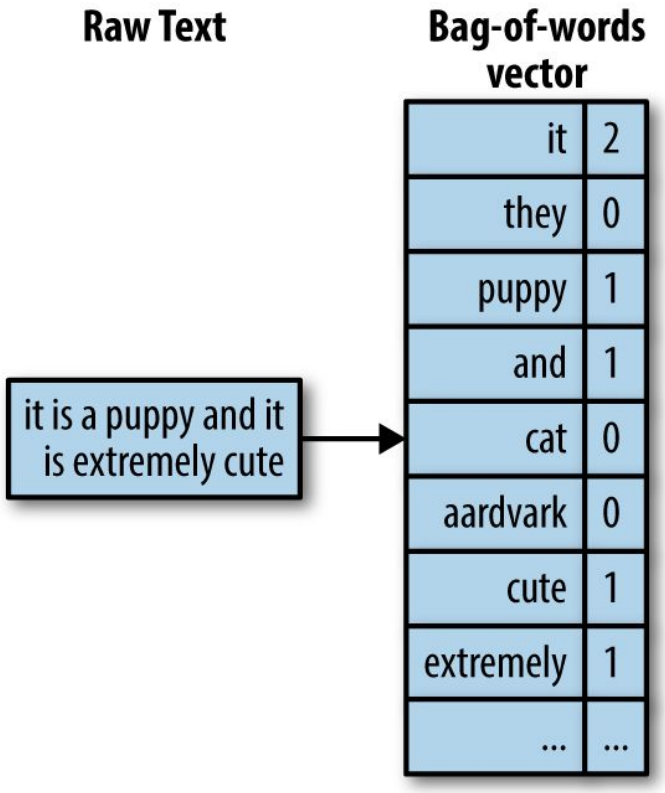
Step 4:
Train Your Model

Text Data



Step 4:
Train Your Model

Text Data



Step 4:
Train Your Model

Text Data

$$w_{x,y} = tf_{x,y} \times \log\left(\frac{N}{df_x}\right)$$

TF-IDF

Term x within document y

$tf_{x,y}$ = frequency of x in y

df_x = number of documents containing x

N = total number of documents

Step 4:
Train Your Model

Sklearn Transformers

1. **Fit**

```
from
sklearn.feature_extraction.text
import TfidfVectorizer
tfidf = TfidfVectorizer()
tfidf.fit_transform(data)
```

2. **Transform**

```
tfidf.transform(new_data)
```

5: Evaluate Model

Step 5: Evaluate Your Model

Model	Accuracy
Shaggy Rogers	80%
Scooby-Doo	92%
Fred Jones	78%
Daphne Blake	85%
Velma Dinkley	79%

```
from sklearn.metrics import
accuracy_score
accuracy_score(actual, predicted)
```

Step 5: Evaluate Your Model

		We predicted it would be...				
		Shaggy Rogers	Scooby-Doo	Fred Jones	Daphne Blake	Velma Dinkley
	Shaggy Rogers	354	8	55	9	54
It actually was	Scooby-Doo	45	147	16	0	7
	Fred Jones	121	14	125	16	93
	Daphne Blake	90	1	66	27	70
	Velma Dinkley	91	4	92	18	153

```
from sklearn.metrics import
confusion_matrix
confusion_matrix(actual, predicted)
```

Step 5: Evaluate Model

Top Features For Shaggy

```
import eli5
eli5.show_weights(model)
```

y=1 top features

Weight[?]	Feature
+5.868	like
+4.449	scoob
+2.517	zoinks
+1.580	us
+1.406	hey
+1.321	groovy
+1.296	buddy
+1.246	we
+1.137	scooby and
+0.962	old
+0.935	stuff
+0.909	vampires
+0.906	mine
… 946 more positive …	
… 1263 more negative …	
-0.930	ben
-1.033	jinkies
-1.091	huh
-1.106	<BIAS>
-1.271	reah
-1.282	yeah
-1.633	shaggy

Step 5: Evaluate Model

Top Features For Scooby

y=1 top features

Weight?	Feature
+3.189	reah
+3.180	yeah
+2.615	huh
+2.280	shaggy
+1.992	shaggy and
+1.921	cyber
+1.369	yikes
+1.366	monster
+1.311	dont know
+1.281	yuck
+1.271	okay
+1.250	oh boy
+1.239	doo
+1.192	dooby doo
… 111 more positive …	
… 2098 more negative …	
-1.196	is
-1.412	to
-1.860	<BIAS>
-1.902	like
-1.983	you
-2.104	the

Step 5: Evaluate Model

Top Features For Velma

y=1 top features

Weight?	Feature
+2.521	jinkies
+1.312	in
+1.286	the
+1.229	of
+1.175	ben
+1.123	these
+1.059	daphne
+1.034	fred
+1.032	already
+1.001	up
+0.997	read
… 908 more positive …	
… 1301 more negative …	
-1.008	zoinks
-1.038	okay
-1.050	velma
-1.081	reah
-1.224	uh
-1.343	yeah
-1.459	<BIAS>
-1.571	scoob
-1.989	like

Step 5: Evaluate Model

Top Features For Fred

y=1 top features

Weight?	Feature
+1.473	our
+1.441	got
+1.316	and scooby
+1.291	yes
+1.207	just
+1.185	gang
+1.130	mom
+1.112	any
+1.016	see
+0.909	velma
+0.884	lets
+0.880	guys
+0.878	uh
…951 more positive …	
…1258 more negative …	
-0.873	jinkies
-0.953	zoinks
-1.036	scooby
-1.114	reah
-1.392	<BIAS>
-1.806	scoob
-2.229	like

Step 5: Evaluate Model

Top Features For Daphne

y=1 top features

Weight?	Feature
+1.691	freddy
+1.390	do
+1.294	you
+1.199	freddie
+1.177	that
+1.155	oh
+1.135	two
+1.071	look
+1.052	does
+0.989	arent
+0.978	hope
+0.937	velma
+0.918	here
…669 more positive …	
…1540 more negative …	
-0.919	shaggy
-0.929	reah
-0.962	got
-1.408	scoob
-1.584	yeah
-1.682	like
-1.784	<BIAS>

Step 5: Evaluate Model

Text Specific Analysis

“Jinkies! What are you doing?”

Predicted: **Velma** (32% likely)

jinkies what are you doing

Very likely to be Velma

Somewhat unlikely to be Velma

Somewhat likely to be Velma

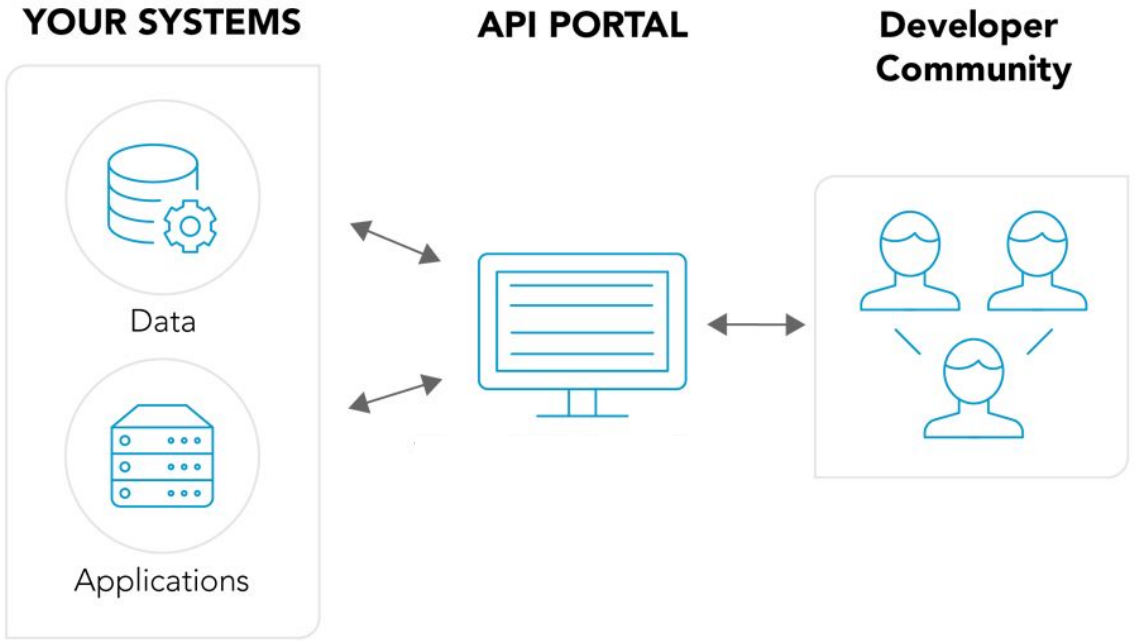
6: Deploy your Model

Step 6:
Deploy Your Model

```
from joblib import dump
dump(model, 'cache/model.joblib')
```

```
from joblib import load
model = load('cache/model.joblib')
```

Step 6:
Deploy Your Model



Step 6:
Deploy Your Model

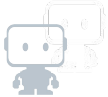
Flask

web development,
one drop at a time

flask.palletsprojects.com/en/1.0.x/quickstart/

```
20 @app.route('/')
21 def index():
22     return render_template('index.html')
23
24 @app.route('/predict/<character>', methods=['POST'])
25 def predict():
26     string = request.json['text']
27     return jsonify(predict_character(string))
```

```python
import numpy as np
from flask import Flask, jsonify, request, render_template

app = Flask(__name__)

def predict_character(text):
    t_text = remove_parentheticals(text)
    t_text = clean_punct(t_text)
    tfidf_text = tfidf.transform([t_text])
    preds = defaultdict(lambda: 0)
    for character in CHARACTERS:
        preds[character] = models[character].predict_proba(tfidf_text)[:, 1][0]
    sumx = sum(preds.values())
    for character in CHARACTERS:
        preds[character] /= sumx
    return {'prediction': list(preds.keys())[np.argmax(list(preds.values()))],
            'probability': np.max(list(preds.values())),
            'probabilities': preds}

@app.route('/')
def index():
    return render_template('index.html')

@app.route('/predict/<character>', methods=['POST'])
def predict():
    string = request.json['text']
    return jsonify(predict_character(string))
```



7: Build Your App

Step 7:
Build Your App

reactjs.org/docs/getting-started.html

Step 7:
Build Your App

```
npx create-react-app my-app
```

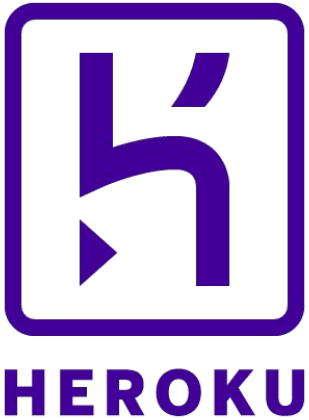
github.com/facebook/create-react-app

Step 7:
Build Your App

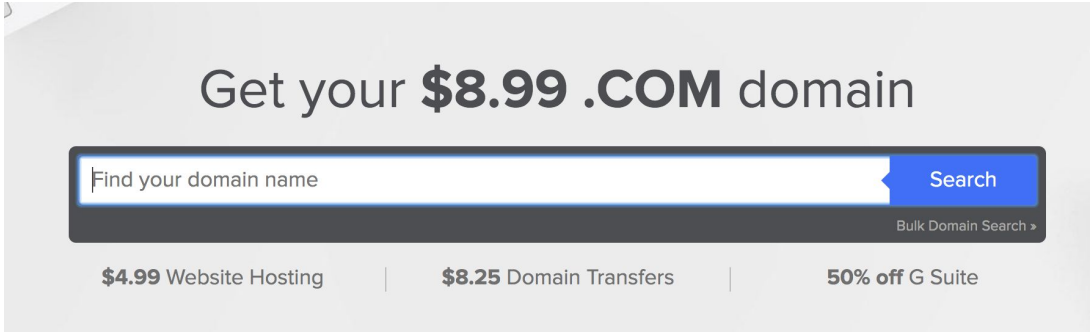
```
function App() {
  const [data, setData] = useState();
  const [textToPredict, setTextToPredict] = useState();

  function predictWhoSaidIt() {
    axios
      .post("http://www.zoinksvsjinkies.com/predict", {
        text: textToPredict
      })
      .then(res => {
        setData(res.data);
        console.log(res);
      });
  }
```

8: Deploy Your App

Step 8:
Deploy your app



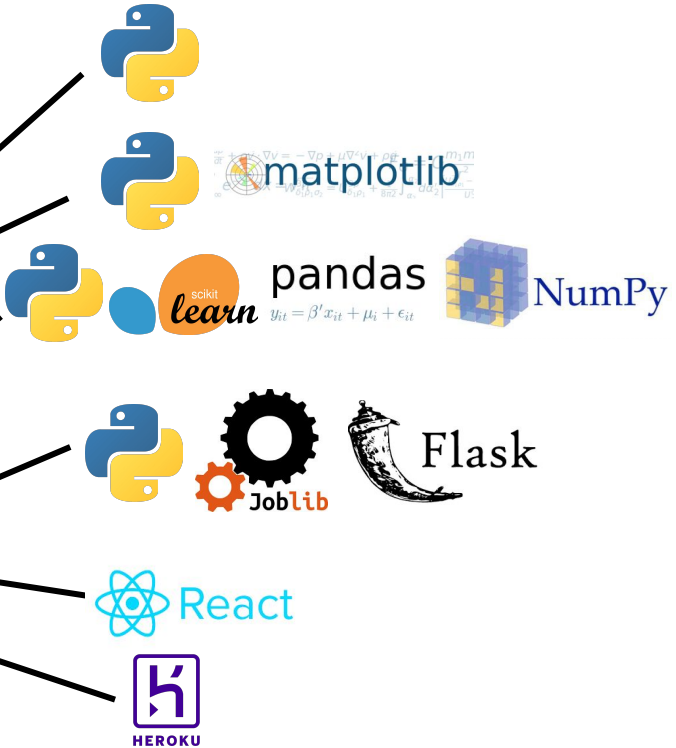
Step 8:
Deploy your app



www.zoinksvsjinkies.com

DATA SCIENCE

1. Define problem
2. Create data set
3. Explore data
4. Train your model
5. Evaluate your model
6. Deploy your model
7. Build your app
8. Deploy your app



Simplify with AI API

Step 8:
Deploy your app

```
import os
from datarobotai.client import DataRobotAIClient

dr = DataRobotAIClient.create(key=os.environ['AI_API_KEY'])
ai = dr.create_ai('Mystery Machine Learning')
ai.learn('character', 'scooby_doo_lines.csv')
prediction = ai.predict('character', [{'line': 'Zoinks!'}])
# [Prediction(0, 'Shaggy Rogers',
#                [{'value': 0.0399286524, 'label': 'Fred Jones'},
#                 {'value': 0.026 6289704, 'label': 'Daphne Blake'},
#                 {'value': 0.0345722802, 'label': 'Velma Dinkley'},
#                 {'value': 0.0506435853, 'label': 'Scooby-Doo'},
#                 {'value': 0.8482265116, 'label': 'Shaggy Rogers'}])]
```

DATA SCIENCE

1. Define problem
2. Create data set
3. Explore data
4. Train your model
5. Evaluate your model
6. Deploy your model
7. Build your app
8. Deploy your app

DataRobot
AI API PRIVATE BETA

85+ lines -> 4 lines of code!

```
dr = DataRobotAIClient.create(key=os.environ['AI_API_KEY'])
ai = dr.create_ai('Mystery Machine Learning')
ai.learn('character', 'scooby_doo_lines.csv')
prediction = ai.predict('character', [{'line': 'Zoinks!'}])
```

developers.datarobot.com

Invite code:
THAT-conference-19

I'm **Peter Hurford**
I work at **DataRobot**

Play with the app: **www.zoinksvsjinkies.com**

Find it on GitHub and explore code:
github.com/peterhurford/mystery_machine_learning

Try the AI API: **developers.datarobot.com**,
use invite code: **THAT-conference-19**

Ask me questions: **peter.hurford@datarobot.com**