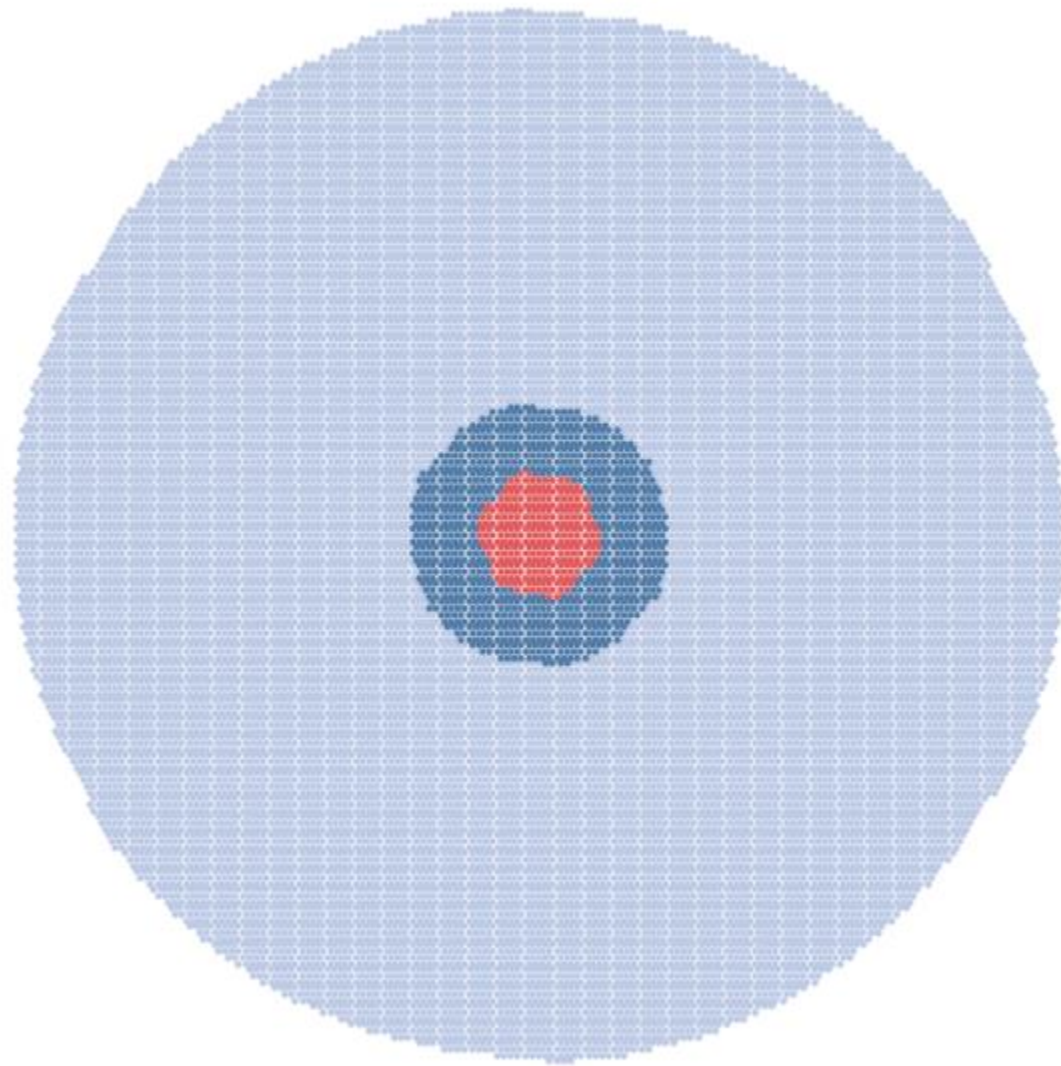


Machine Learning and the Hall of Fame

Cale Green, Camille Goodwin, Claire Davis, Matt Martin, Silas Cobb, Trey Wehrmeyer

HOF Induction Status-Circle



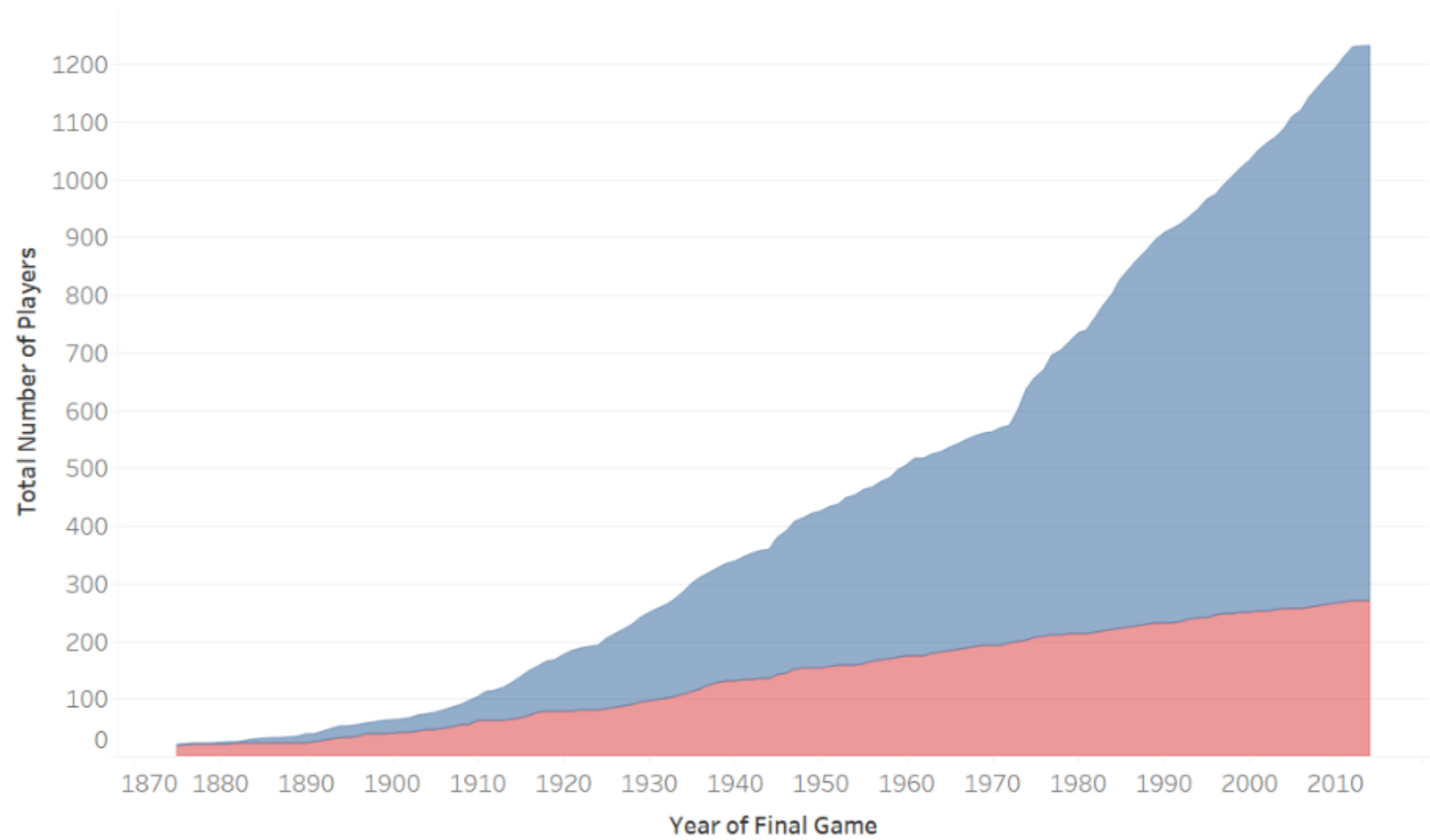
Inducted

Y

N

Not Eligible

Total Players vs. HOF Induction Status



Inducted

N

Y

Proposal

Using machine learning to predict future members to the Baseball Hall of Fame

- Predict future inductees
- Examine the statistically-worthy players who have not yet been inducted
- Recognize historically underrated players

Resources

Sean Lahman's Baseball Database

Baseball Reference

Baseball Hall

Fangraphs

Tables Used:

- Pitching.csv
- Batting.csv
- HallOfFame.csv

For additional information and resources: [Click Here!](#)

Technologies Used

- Python and Pandas for data exploration and ETL
- PostgreSQL for a database
- Extreme Gradient Boosting machine learning model
- Matplotlib and Tableau for visualization of findings
- Heroku, Flask, HTML, and JavaScript for dashboard creation and deployment

ETL

Batting ETL:

Created Career_Batting_df by:

- Merging Batting and Hall of Fame CSV's
- Dropping columns with too many null values
- Adding some necessary calculated statistics

Pitching ETL:

Created Career_Pitching_df by:

- Merging the pitching and Hall of Fame CSV's
- Dropping players who did not meet the minimum time to be eligible for the HOF
- Additional necessary calculated columns were added

Final Hall Batter Dataframe in Jupyter Notebook

```
# Setting playerID as index
```

```
hall_batter_df = hall_batter_df.set_index('playerID')  
hall_batter_df
```

	G	AB	R	H	2B	3B	HR	RBI	SB	BB	SO	HBP	SH	SF	AVG	OBP	SLG	inducted
playerID																		
aardsda01	331.0	4.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	2.0	0.0	1.0	0.0	0.000000	0.000000	0.000000	N
aaronha01	3298.0	12364.0	2174.0	3771.0	624.0	98.0	755.0	2297.0	240.0	1402.0	1383.0	32.0	21.0	121.0	0.304998	0.373949	0.554513	Y
aaronto01	437.0	944.0	102.0	216.0	42.0	6.0	13.0	94.0	9.0	86.0	145.0	0.0	9.0	6.0	0.228814	0.291506	0.327331	N
aasedo01	448.0	5.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	3.0	0.0	0.0	0.0	0.000000	0.000000	0.000000	N
abadan01	15.0	21.0	1.0	2.0	0.0	0.0	0.0	0.0	0.0	4.0	5.0	0.0	0.0	0.0	0.095238	0.240000	0.095238	N
...
zupcibo01	319.0	795.0	99.0	199.0	47.0	4.0	7.0	80.0	7.0	57.0	137.0	6.0	20.0	8.0	0.250314	0.302540	0.345912	N
zupofr01	16.0	18.0	3.0	3.0	1.0	0.0	0.0	0.0	0.0	2.0	6.0	0.0	0.0	0.0	0.166667	0.250000	0.222222	N
zuvelpa01	209.0	491.0	41.0	109.0	17.0	2.0	2.0	20.0	2.0	34.0	50.0	2.0	18.0	0.0	0.221996	0.275142	0.276986	N
zuverge01	266.0	142.0	5.0	21.0	2.0	1.0	0.0	7.0	0.0	9.0	39.0	0.0	16.0	0.0	0.147887	0.198675	0.176056	N
zwilldu01	366.0	1280.0	167.0	364.0	76.0	15.0	30.0	202.0	46.0	128.0	155.0	4.0	31.0	0.0	0.284375	0.351275	0.437500	N

Database

- Create connection to database from ETL documents
- Import 4 tables to PostgreSQL
- Pull tables from Postgres to machine learning documents with cursor object

Database Connection

```
In [22]: hall_batter_df.to_csv('./Resources/Revised_CSV/hall_batter.csv')
        career_batter_df.to_csv('./Resources/Revised_CSV/career_batter.csv')
```

```
In [22]: # import create_engine from the sqlalchemy module
        from sqlalchemy import create_engine
```

```
In [23]: # import password from config file
        from config import db_password
```

```
In [24]: # connection string
        db_string = f"postgres://postgres:{db_password}@127.0.0.1:5432/baseball_data"
```

```
In [25]: # create database engine
        engine = create_engine(db_string)
```

```
In [26]: # import tables
        hall_batter_df.to_sql(name='hall_batter', con=engine, if_exists='replace')
        career_batter_df.to_sql(name='career_batter', con=engine, if_exists='replace')
```

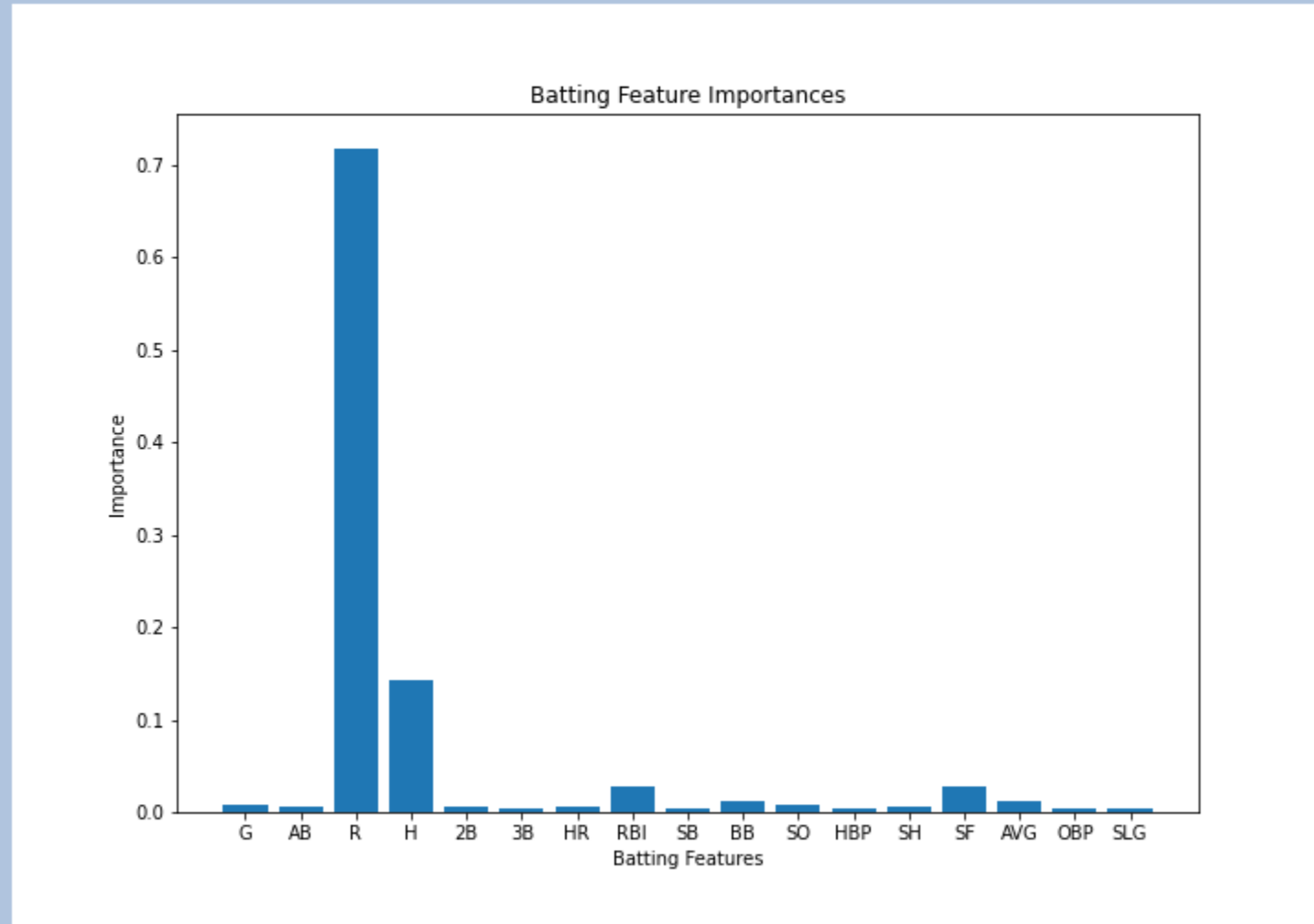
Live Demo

<https://hall-of-fame-baseball.herokuapp.com/>

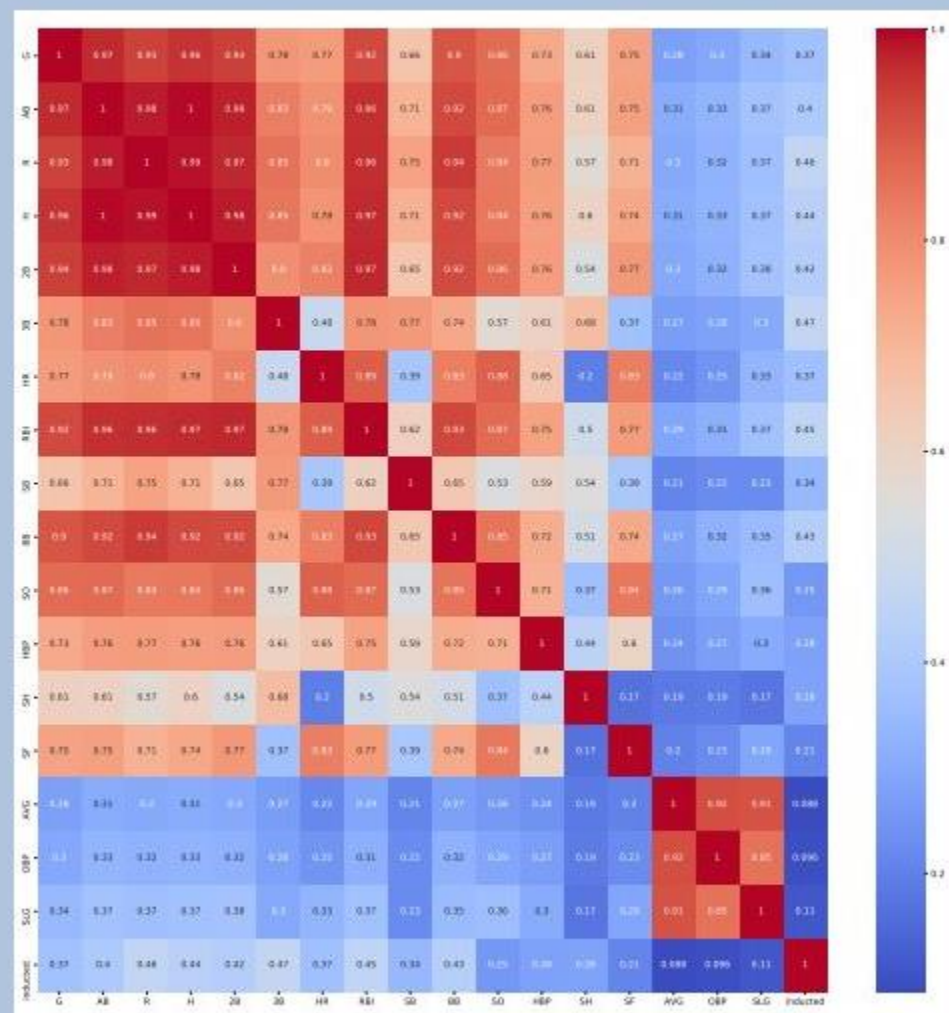
Machine Learning Model

- Targeted models with high recall scores
 - Finding eligible players from so many ineligible
- Secondary priority, precision scores
 - If predicted in HOF, how likely is it true

Feature Importances



Correlation Between Batting Features



Comparison of Tested Models

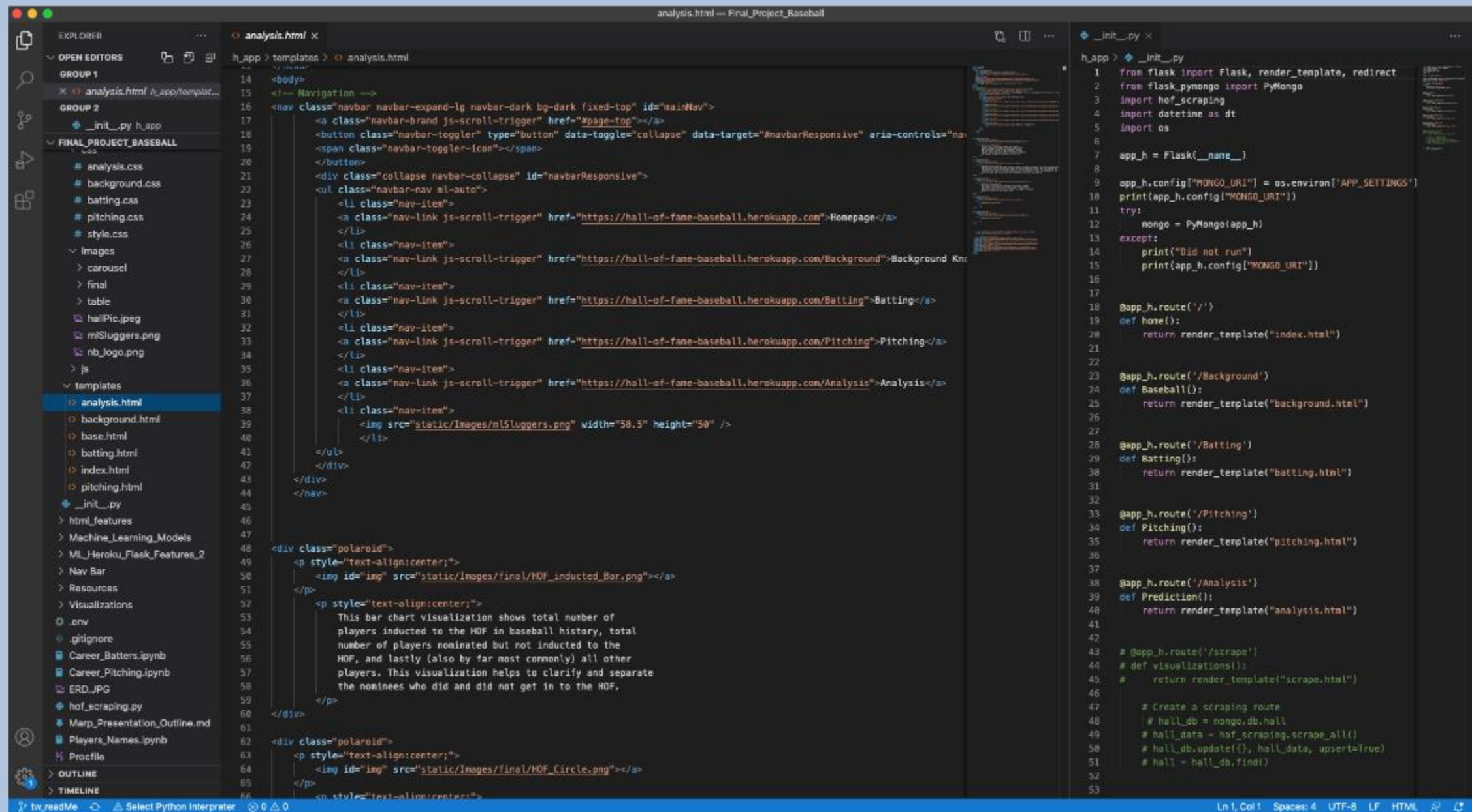
Model and Sampling Method	Accuracy	Yes/No	Precision	Recall	F-1 Score
<i>XGBoost Classifier with SMOTE</i>	98.9%	No	1.00	0.99	0.99
		Yes	0.46	0.77	0.58
<i>Random Forest Classifier (RFC)</i>	99.2%	No	0.99	1.00	1.00
		Yes	0.63	0.44	0.52
<i>RFC with SMOTE</i>	98.5%	No	1.00	0.99	0.99
		Yes	0.31	0.76	0.44

	Name	G	AB	R	H	2B	3B	HR	RBI	SB	BB	SO	HBP	SH	SF	AVG	OBP	SLG	HOF Prediction	Yes HOF Probability
0	Barry Bonds	2986	9847	2227	2935	601	77	762	1996	514	2558	1539	106	4	91	0.298000	0.444000	0.607000	Y	99.02%
1	Manny Ramirez	2302	8244	1544	2574	547	20	555	1831	38	1329	1813	109	2	90	0.312000	0.411000	0.585000	Y	63.03%
2	Todd Helton	2247	7962	1401	2519	592	37	369	1406	37	1335	1175	57	3	93	0.316000	0.414000	0.539000	N	9.08%
3	Alex Rodriguez	2784	10566	2021	3115	548	31	696	2086	329	1338	2287	176	16	111	0.295000	0.380000	0.550000	Y	55.80%
4	David Ortiz	2408	8640	1419	2472	632	19	541	1768	17	1319	1750	38	2	92	0.286000	0.380000	0.552000	N	1.16%
5	Gary Sheffield	2576	9217	1636	2689	467	27	509	1676	253	1475	1171	135	9	111	0.292000	0.393000	0.514000	Y	99.95%
6	Bobby Abreu	2425	8480	1453	2470	574	59	288	1363	400	1476	1840	33	7	85	0.291000	0.395000	0.475000	N	2.88%
7	Prince Fielder	1611	5821	862	1645	321	10	319	1028	18	847	1155	124	0	61	0.283000	0.382000	0.506000	N	0.01%
8	Mark Teixeira	1862	6936	1099	1862	408	18	409	1298	26	918	1441	111	0	64	0.268000	0.360000	0.509000	N	0.23%
9	Sammy Sosa	2354	8813	1475	2408	379	45	609	1667	234	929	2306	59	17	78	0.273000	0.344000	0.534000	Y	57.20%
10	Scott Rolen	2038	7398	1211	2077	517	43	316	1287	118	899	1410	127	1	93	0.281000	0.364000	0.490000	N	0.17%
11	Jeff Kent	2298	8498	1320	2461	560	47	377	1518	94	801	1522	125	10	103	0.290000	0.356000	0.500000	N	33.14%
12	Ryan Howard	1572	5707	848	1475	277	21	382	1194	12	709	1843	59	0	55	0.258000	0.343000	0.515000	N	0.02%
13	Justin Morneau	1545	5699	772	1603	349	23	247	985	5	573	988	46	0	74	0.281000	0.348000	0.481000	N	0.01%
14	Andruw Jones	2196	7599	1204	1933	383	36	434	1289	152	891	1748	97	6	71	0.254000	0.337000	0.486000	N	1.18%
15	Billy Butler	1414	5105	592	1479	322	5	147	728	5	500	840	34	0	47	0.290000	0.354000	0.441000	N	0.00%
16	Torii Hunter	2372	8857	1296	2452	498	39	353	1391	195	661	1741	97	6	71	0.277000	0.331000	0.461000	N	14.59%
17	Carl Crawford	1716	6655	998	1931	309	123	136	766	480	377	1067	47	34	48	0.290000	0.330000	0.435000	N	0.00%
18	Marlon Byrd	1574	5579	740	1534	311	39	159	710	56	382	1234	93	17	52	0.275000	0.329000	0.430000	N	0.00%
19	Jimmy Rollins	2275	9294	1421	2455	511	115	231	936	470	813	1264	38	42	53	0.264000	0.324000	0.418000	N	4.58%
20	Angel Pagan	1124	4084	584	1143	220	55	64	414	176	319	638	4	21	34	0.280000	0.330000	0.408000	N	0.00%
21	Coco Crisp	1586	5930	877	1572	308	57	130	639	309	561	865	5	80	51	0.265000	0.327000	0.402000	N	0.00%
22	A.J. Pierzynski	2059	7290	807	2043	407	24	188	909	15	308	895	129	28	58	0.280000	0.319000	0.420000	N	0.00%
23	Juan Uribe	1826	6161	724	1568	323	43	199	816	48	388	1224	46	63	57	0.255000	0.301000	0.418000	N	0.00%
24	Omar Vizquel	2968	10586	1445	2877	456	77	80	951	404	1028	1087	49	256	94	0.272000	0.336000	0.352000	Y	79.81%
25	Michael Bourn	1361	4784	678	1272	191	69	36	361	341	445	1124	19	52	23	0.266000	0.329000	0.357000	N	0.00%
26	Omar Infante	1507	5271	609	1427	260	53	82	542	80	296	776	14	64	53	0.271000	0.308000	0.387000	N	0.00%

Deployment and Heroku

- Used HTML, CSS, Bootstrap, JS, Python, and Flask to build dynamic webpage
- Ability to scrape HOF website top article and pictures (Currently local feature)
- Set up file system according to Heroku's specifications to deploy our project
 - h_app, html files, Procfile, requirements, wsgi

NavBar Breakdown



```
analysis.html
14 <body>
15 <!-- Navigation -->
16 <nav class="navbar navbar-expand-lg navbar-dark bg-dark fixed-top" id="mainNav">
17   <a class="navbar-brand js-scroll-trigger" href="#page-top"></a>
18   <button class="navbar-toggler" type="button" data-toggle="collapse" data-target="#navbarResponsive" aria-controls="navbarResponsive" aria-expanded="false" aria-label="Toggle navigation">
19     <span class="navbar-toggler-icon"></span>
20   </button>
21   <div class="collapse navbar-collapse" id="navbarResponsive">
22     <ul class="navbar-nav ml-auto">
23       <li class="nav-item">
24         <a class="nav-link js-scroll-trigger" href="https://hall-of-fame-baseball.herokuapp.com/>Homepage</a>
25       </li>
26       <li class="nav-item">
27         <a class="nav-link js-scroll-trigger" href="https://hall-of-fame-baseball.herokuapp.com/Background">Background Knowledge</a>
28       </li>
29       <li class="nav-item">
30         <a class="nav-link js-scroll-trigger" href="https://hall-of-fame-baseball.herokuapp.com/Batting">Batting</a>
31       </li>
32       <li class="nav-item">
33         <a class="nav-link js-scroll-trigger" href="https://hall-of-fame-baseball.herokuapp.com/Pitching">Pitching</a>
34       </li>
35       <li class="nav-item">
36         <a class="nav-link js-scroll-trigger" href="https://hall-of-fame-baseball.herokuapp.com/Analysis">Analysis</a>
37       </li>
38       <li class="nav-item">
39         
40       </li>
41     </ul>
42   </div>
43 </nav>
44
45 <div class="polaroid">
46   <p style="text-align:center;">
47     
48   </p>
49   <p style="text-align:center;">
50     This bar chart visualization shows total number of
51     players inducted to the HOF in baseball history, total
52     number of players nominated but not inducted to the
53     HOF, and lastly (also by far most commonly) all other
54     players. This visualization helps to clarify and separate
55     the nominees who did and did not get in to the HOF.
56   </p>
57 </div>
58
59 <div class="polaroid">
60   <p style="text-align:center;">
61     
62   </p>
63   <p style="text-align:center;">
64     This circle chart visualization shows the percentage of
65     players inducted to the HOF in baseball history, total
66     number of players nominated but not inducted to the
67     HOF, and lastly (also by far most commonly) all other
68     players. This visualization helps to clarify and separate
69     the nominees who did and did not get in to the HOF.
70   </p>
71 </div>
72
73 </body>
74 </html>
```

```
_init_.py
1 from flask import Flask, render_template, redirect
2 from flask_pymongo import PyMongo
3 import hof_scraping
4 import datetime as dt
5 import os
6
7 app = Flask(__name__)
8
9 app.config["MONGO_URI"] = os.environ['APP_SETTINGS']
10 print(app.config["MONGO_URI"])
11 try:
12     mongo = PyMongo(app)
13 except:
14     print("Did not run")
15     print(app.config["MONGO_URI"])
16
17 @app.route('/')
18 def home():
19     return render_template("index.html")
20
21 @app.route('/Background')
22 def Background():
23     return render_template("background.html")
24
25 @app.route('/Batting')
26 def Batting():
27     return render_template("batting.html")
28
29 @app.route('/Pitching')
30 def Pitching():
31     return render_template("pitching.html")
32
33 @app.route('/Analysis')
34 def Prediction():
35     return render_template("analysis.html")
36
37 # @app.route('/scrape')
38 # def visualizations():
39 #     return render_template("scrape.html")
40
41 # Create a scraping route
42 # hall_db = mongo.db.hall
43 # hall_data = hof_scraping.scrape_all()
44 # hall_db.update({}, hall_data, upsert=True)
45 # hall = hall_db.find()
```

Code to Filter Interactive Table

```
function filterTable() {  
  let filteredBatterData = batterData;  
  Object.entries(filters).forEach(([key, value]) => {  
    filteredBatterData = filteredBatterData.filter(row => row[key] === value);  
  });  
  buildTable(filteredBatterData);  
}  
  
d3.selectAll("input").on("change", updateFilters);  
  
buildTable(batterData);
```

Future Advancements

- Expand to other sports or awards
- Predict all star teams
- More interactive pieces - prediction table to enter in stats and see probability of being inducted
- Work on style CSS to be more user friendly and clean

Thank you for your attention.

Any questions or concerns?