Song Popularity & Algorithm Comparison

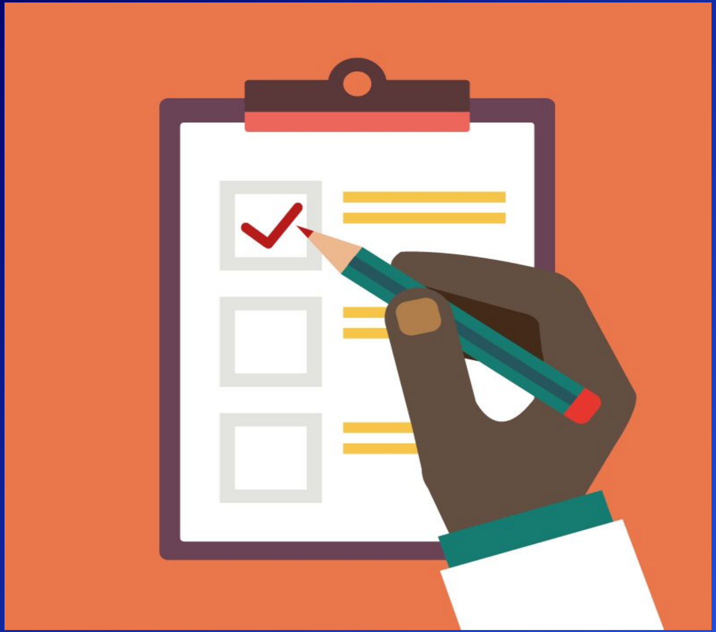
Osher Boudara

09/28/22

Cognizant

Agenda

1. About Me
2. Motivation
3. Business/Technical Impact
4. Project Pipeline
5. Dataset
6. Data Preprocessing
7. Data Scaling
8. Algorithms
9. Results
10. Conclusion/Future Research



Cognizant

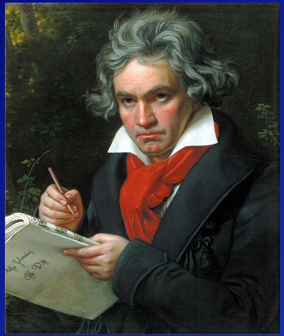
About Me: Osher Boudara

- Role
  - Data Engineer, Associate
- Interests
  - Machine Learning
  - Data Warehousing
  - Python Shorthand Techniques
- Fun Fact
  - Spent a year studying abroad in Israel after high school

Motivation

- Throughout history, music was only heard if played live or if one knew how to play.
  - Select few can classify what a popular or good song is.
- In the modern day, music streaming services allow us to listening to almost anything we want at the tip of our fingers.
  - Anyone can classify a popular or good song by simply listening to it on their mobile device.
- Because of this, it is my interest to determine what makes a song popular.

Popular Then



VS.

Popular Now

Cognizant

Business Impact

- Musicians and record labels alike can utilize the certain features of a song to help create songs that will become popular.
- Further, using current data to build prediction models will give us insight into whether a new song created can be popular.

Short Term Impact	Long Term Impact
Musicians can produce a popular song by utilizing certain song features over others.	Musicians and record labels can construct a system for using more favorable features to consistently produce popular songs.

Cognizant

Technical Impact

- In order to make valid predictions on whether a song is popular, it is important to figure out the best performing model.
- A few algorithms were used to learn what makes a song popular:
    - Linear Regression
    - XGBoost Regressor
    - Random Forest Regressor
- Algorithm metrics will be assessed to determine best performing model.

Cognizant

Project Pipeline

Dataset

- Spotify Dataset from Kaggle
- Dataset Shape:
    - Rows: 232725
    - Columns: 18
- *Popularity*
    - Target Feature
    - Value between 0-100 before scaling

Popularity Max	Popularity Min
100	0

Features of Spotify Dataset

```
---------------------------
Artist_Name
Track_Name
Track_Id
Genre
Key
Mode
Time_Signature
Acousticness
Danceability
Duration_Ms
Energy
Instrumentalness
Liveness
Loudness
Speechiness
Tempo
Valence
Popularity
```

Informational Data

Categorical Data

Numerical Data

Dataset

	genre	artist_name	track_name	track_id	popularity	acousticness	danceability	duration_ms	energy	instrumentalness	key	liveness	loudness	mode	speechiness	tempo	time_signature	valence
0	Movie	Henri Salvador	C'est beau de faire un Show	0BRjO6ga9RKCKjfDqeFgWV	0	0.611	0.389	99373	0.910	0.000	C#	0.3460	-1.828	Major	0.0525	166.969	4/4	0.814
1	Movie	Martin & les fées	Perdu d'avance (par Gad Elmaleh)	0BjC1NfoEOOusryehmNudP	1	0.246	0.590	137373	0.737	0.000	F#	0.1510	-5.559	Minor	0.0868	174.003	4/4	0.816
2	Movie	Joseph Williams	Don't Let Me Be Lonely Tonight	0CoSDzoNIKCRs124s9uTVy	3	0.952	0.663	170267	0.131	0.000	C	0.1030	-13.879	Minor	0.0362	99.488	5/4	0.368
3	Movie	Henri Salvador	Dis-moi Monsieur Gordon Cooper	0Gc6TVm52BwZD07Ki6tIvf	0	0.703	0.240	152427	0.326	0.000	C#	0.0985	-12.178	Major	0.0395	171.758	4/4	0.227
4	Movie	Fabien Nataf	Ouverture	0IusIXpMROHdEPvSI1fTQK	4	0.950	0.331	82625	0.225	0.123	F	0.2020	-21.150	Major	0.0456	140.576	4/4	0.390

kaggle™

Cognizant

Data Preprocessing

- Genre column had two variations of Children’s Music value. This was fixed to only have one value.
- Check for null values
  - Null values did not exist in the dataset.

```python
#Check for any obscure genres, children's music had two genres
file.genre.unique()
```

```
array(['Movie', 'R&B', 'A Capella', 'Alternative', 'Country', 'Dance',
       'Electronic', 'Anime', 'Folk', 'Blues', 'Opera', 'Hip-Hop',
       "Children's Music", 'Children’s Music', 'Rap', 'Indie',
       'Classical', 'Pop', 'Reggae', 'Reggaeton', 'Jazz', 'Rock', 'Ska',
       'Comedy', 'Soul', 'Soundtrack', 'World'], dtype=object)
```

```python
new_df = file.copy()
```

```python
# Place childrens music different columns under same column
new_df['genre'] = file['genre'].replace(['Children\'s Music'], 'Children’s Music')
```

```python
new_df.genre.unique()
```

```
array(['Movie', 'R&B', 'A Capella', 'Alternative', 'Country', 'Dance',
       'Electronic', 'Anime', 'Folk', 'Blues', 'Opera', 'Hip-Hop',
       'Children’s Music', 'Rap', 'Indie', 'Classical', 'Pop', 'Reggae',
       'Reggaeton', 'Jazz', 'Rock', 'Ska', 'Comedy', 'Soul', 'Soundtrack',
       'World'], dtype=object)
```

```python
# Check for null values
new_df.isnull().sum()
```

```
genre               0
artist_name         0
track_name          0
track_id            0
popularity          0
acousticness        0
danceability        0
duration_ms         0
energy              0
instrumentalness    0
key                 0
liveness            0
loudness            0
mode                0
speechiness         0
tempo               0
time_signature      0
valence             0
dtype: int64
```

Data Preprocessing

- Checked for duplicate songs
  - Duplicate songs had varying genre labels and popularity values.
  - Duplicate songs were kept to assess if genre played a part into popularity.
- Dropped informational columns
  - Columns dropped: *artist_name, track_name, track_id*

```
#   Column
--  ------
0   genre
1   popularity
2   acousticness
3   danceability
4   duration_ms
5   energy
6   instrumentalness
7   key
8   liveness
9   loudness
10  mode
11  speechiness
12  tempo
13  time_signature
14  valence
```

```
# Check if same songs but labeled as different genre have different popularity
df.filter((df.artist_name == 'System Of A Down') & (df.track_name == 'Chop Suey!')).show()

# they do have different popularity so genre label can be integral for popularity, did not remove
# duplicate songs for that reason
```

genre	artist_name	track_name	track_id	popularity	acousticness	danceability	duration_ms	energy	instrumentalness	key	liveness	loudness	mode	speechiness	tempo	time_signature	valence
Alternative	System Of A Down	Chop Suey!	2DlHlPMa4M17kufBv...	77	2.78E-4	0.419	210240	0.934	0.0015	G	0.132	-3.908	Minor	0.12	127.288	4/4	0.286
Children’s Music	System Of A Down	Chop Suey!	2DlHlPMa4M17kufBv...	80	2.78E-4	0.419	210240	0.934	0.0015	G	0.132	-3.908	Minor	0.12	127.288	4/4	0.286
Rap	System Of A Down	Chop Suey!	2DlHlPMa4M17kufBv...	81	2.78E-4	0.419	210240	0.934	0.0015	G	0.132	-3.908	Minor	0.12	127.288	4/4	0.286
Rock	System Of A Down	Chop Suey!	2DlHlPMa4M17kufBv...	81	2.78E-4	0.419	210240	0.934	0.0015	G	0.132	-3.908	Minor	0.12	127.288	4/4	0.286

Cognizant

Data Scaling: Numerical

- Numerical data was scaled using Min Max Scaler
  - MinMaxScaler scales values between 0 and 1
- Scaling was done using PySpark VectorAssembler and a pipeline.
- After transforming dataset to contain original values, vector values and scaled values, original and vector values were dropped while scaled values were kept in order to build the model.

```
# Create vectors of column value and placed into column with new name
# Create scaled column vectors and placed into new column with new name
# Pass through pipeline transformation to achieve scaled values and put all columns
# (Vector, Unscaled, Scaled vectors) into one spark dataframe
assemblers = [VectorAssembler(inputCols=[col], outputCol= col + '_vec') for col in numerical_columns]
scalers = [MinMaxScaler(inputCol=col+'_vec', outputCol=col + '_scaled') for col in numerical_columns]

pipeline = Pipeline(stages=assemblers + scalers)
model_min_max_scaler = pipeline.fit(df_numeric)
df_numeric_scaled = model_min_max_scaler.transform(df_numeric)
```

	acousticness	danceability	duration_ms	energy	instrumentalness	liveness	loudness	speechiness	tempo	valence	popularity
0	0.613454	0.356292	0.015167	0.910909	0.000000	0.339614	0.900856	0.032070	0.642704	0.814	0.00
1	0.246988	0.571934	0.022029	0.737732	0.000000	0.142710	0.834469	0.068374	0.675801	0.816	0.01
2	0.955823	0.650252	0.027969	0.131113	0.000000	0.094241	0.686429	0.014818	0.325182	0.368	0.03
3	0.705823	0.196438	0.024747	0.326313	0.000000	0.089697	0.716695	0.018311	0.665238	0.227	0.00
4	0.953815	0.294067	0.012142	0.225209	0.123123	0.194208	0.557054	0.024767	0.518516	0.390	0.04
...	...	...	...	...	...	...	...	...	...	...	...
232720	0.003855	0.676000	0.056136	0.714709	0.544545	0.075561	0.744311	0.009949	0.400722	0.962	0.39
232721	0.033032	0.781139	0.048227	0.683677	0.000881	0.229550	0.809825	0.012172	0.392666	0.969	0.38
232722	0.904619	0.493617	0.027372	0.419408	0.000000	0.085658	0.786018	0.133150	0.252941	0.813	0.47
232723	0.263052	0.738226	0.037391	0.704699	0.000000	0.326487	0.806391	0.131033	0.327737	0.489	0.44
232724	0.097691	0.752173	0.055555	0.470460	0.000049	0.074652	0.814025	0.006880	0.392981	0.479	0.35

232725 rows × 11 columns

Cognizant

Data Scaling: Categorical

- Categorical data was handled using One Hot Encoding.
- This was accomplished using the Pandas *get_dummies* function.
- Columns subject to One Hot Encoding are:
  - *genre*
  - *key*
  - *mode*
  - *time_signature*

Color
Red
Red
Yellow
Green
Yellow

Red	Yellow	Green
1	0	0
1	0	0
0	1	0
0	0	1

#	Column
0	genre_A Capella
1	genre_Alternative
2	genre_Anime
3	genre_Blues
4	genre_Children's Music
5	genre_Classical
6	genre_Comedy
7	genre_Country
8	genre_Dance
9	genre_Electronic
10	genre_Folk
11	genre_Hip-Hop
12	genre_Indie
13	genre_Jazz
14	genre_Movie
15	genre_Opera
16	genre_Pop
17	genre_R&B
18	genre_Rap
19	genre_Reggae
20	genre_Reggaeton
21	genre_Rock
22	genre_Ska
23	genre_Soul
24	genre_Soundtrack
25	genre_World
26	key_A
27	key_A#
28	key_B
29	key_C
30	key_C#
31	key_D
32	key_D#
33	key_E
34	key_F
35	key_F#
36	key_G
37	key_G#
38	mode_Major
39	mode_Minor
40	time_signature_0/4
41	time_signature_1/4
42	time_signature_3/4
43	time_signature_4/4
44	time_signature_5/4

Algorithms: Initial Steps and Linear Regression

- Correlation coefficients between popularity and other numeric features were assessed to see if any one feature determines popularity.
  - The correlation coefficients were low (less than 0.5 and greater than -0.5).
  - This fueled the idea for prediction models to see how collectively each feature plays into the popularity value.
- Linear Regression
  - Approach for modeling relationship between scalar variable (target) and one or more explanatory variables.

	popularity	acousticness	danceability	duration_ms	energy	instrumentalness	liveness	loudness	speechiness	tempo	valence
popularity	1.000000	-0.381295	0.256564	0.002348	0.248922	-0.210983	-0.167995	0.363011	-0.151076	0.081039	0.060076
acousticness	-0.381295	1.000000	-0.364546	0.011203	-0.725576	0.316154	0.069004	-0.690202	0.150935	-0.238247	-0.325798
danceability	0.256564	-0.364546	1.000000	-0.125781	0.325807	-0.364941	-0.041684	0.438668	0.134560	0.021939	0.547154
duration_ms	0.002348	0.011203	-0.125781	1.000000	-0.030550	0.076021	0.023783	-0.047618	-0.016171	-0.028456	-0.141811
energy	0.248922	-0.725576	0.325807	-0.030550	1.000000	-0.378957	0.192801	0.816088	0.145120	0.228774	0.436771
instrumentalness	-0.210983	0.316154	-0.364941	0.076021	-0.378957	1.000000	-0.134198	-0.506320	-0.177147	-0.104133	-0.307522
liveness	-0.167995	0.069004	-0.041684	0.023783	0.192801	-0.134198	1.000000	0.045686	0.510147	-0.051355	0.011804
loudness	0.363011	-0.690202	0.438668	-0.047618	0.816088	-0.506320	0.045686	1.000000	-0.002273	0.228364	0.399901
speechiness	-0.151076	0.150935	0.134560	-0.016171	0.145120	-0.177147	0.510147	-0.002273	1.000000	-0.081541	0.023842
tempo	0.081039	-0.238247	0.021939	-0.028456	0.228774	-0.104133	-0.051355	0.228364	-0.081541	1.000000	0.134857
valence	0.060076	-0.325798	0.547154	-0.141811	0.436771	-0.307522	0.011804	0.399901	0.023842	0.134857	1.000000

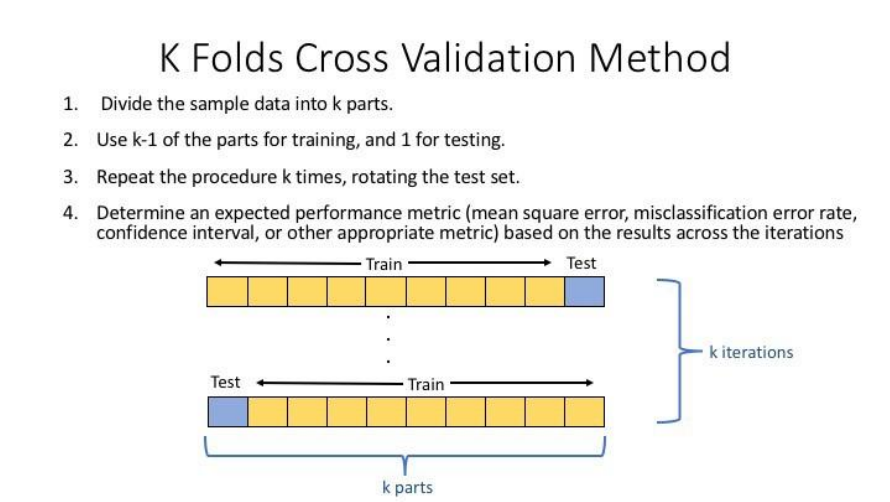
Algorithms: RF and XGB Regressors

- Random Forest Regressor:
  - “A meta estimator that fits a number of classifying decision trees on various sub-samples of the dataset and uses averaging to improve the predictive accuracy and control over-fitting.” [1]
  - The latter portion of the above statement is what differs random forest classification from random forest regressor.
  - The model built had was given a max_depth value of 2.
- XGBoost Regressor:
  - Efficient implementation of the gradient boosting algorithm.
  - Gradient boosting is a class of ensemble machine learning algorithms that can be used for regression predictive modeling problems.

dmlc XGBoost

Algorithms: Additional Information

- For splitting the data, repeated k-fold cross validation was used in order to obtain the best possible model of each of the previously discussed models.
- For all the algorithms, k was set equal to 5 and there were 2 repeats.
  - This means each model was implemented 10 times to obtain best model from each respective algorithm.



Results: Algorithm Comparison

Linear Regression

```
Metrics for best performing Linear Regression model
---------------------------------------------------
MSE: 0.012348584
RMSE: 0.11112418
R^2: 0.6287180211686056
---------------------------------------------------
CPU times: user 7.5 s, sys: 1.42 s, total: 8.92 s
Wall time: 5.41 s
```

Fastest

Random Forest Reg.

```
Metrics for best performing Random Forest model
-----------------------------------------------
MSE: 0.025081842191173359
RMSE: 0.15837247927330553
R^2: 0.24044417852733324
-----------------------------------------------
CPU times: user 5min 46s, sys: 599 ms, total: 5min 46s
Wall time: 5min 48s
```

Worst

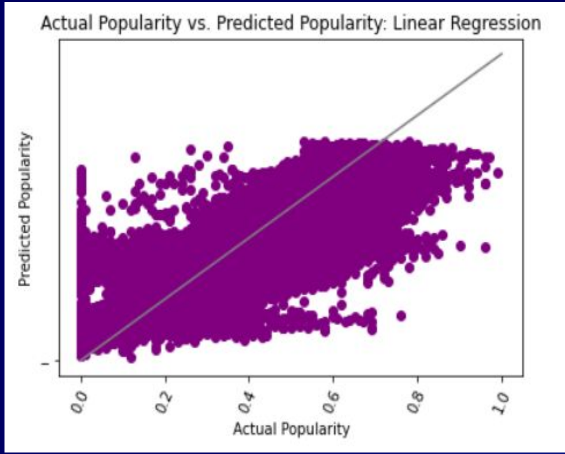
XGBoost Regressor

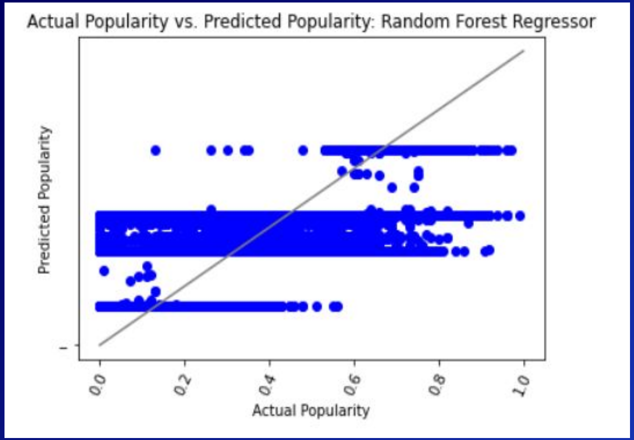
```
Metrics for best performing XGBoost model
-----------------------------------------
MSE: 0.011454776
RMSE: 0.10702699
R^2: 0.6521474798156409
-----------------------------------------
CPU times: user 4min 54s, sys: 685 ms, total: 4min 54s
Wall time: 4min 57s
```

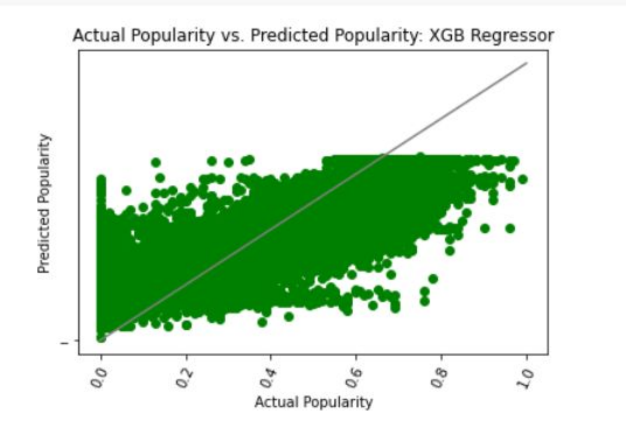
Best

Cognizant

Results: Algorithm Comparison







Cognizant

Results: Business Insight

- Time Signatures and Keys were weighted highest.
    - This means that as these independent variables increase, the dependant variable (Popularity) will increase.
- Therefore, musicians and record labels should focus on these features over others.
- Using our best performing model, musicians and record labels can determine whether a song will be popular.

category	
time_signature_0/4	170.170151
time_signature_4/4	170.128448
time_signature_5/4	170.125000
time_signature_3/4	170.119629
time_signature_1/4	170.117004
key_F#	40.219810
key_G#	40.216576
key_C#	40.215973
key_B	40.214645
key_D#	40.213943
key_A#	40.210503
key_F	40.210052
key_E	40.209595
key_A	40.209465
key_D	40.209198
key_G	40.209152
key_C	40.208363
duration_ms	0.219493
loudness	0.103868
energy	0.029150
tempo	-0.015553
danceability	-0.018230

Cognizant

Conclusion

- Using certain features over others, musicians and record labels can produce songs that will become popular.
- Features such as time signature, key should be focused on more according to weights.
- Best performing model for song popularity predictions:
  - XGBoost Regressor
- The best performing model can help musicians determine if a song will become popular.

Cognizant

Future Research

- Investigation of other regression algorithms that may perform quicker and have better metrics.
- Access Spotify API to analyze data in real time.
    - Use Kafka to stream real time data.
- Acquire more features that may contribute to song popularity.
    - Features such as year of release.

Cognizant

Thank you.

Cognizant

References

- https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestRegressor.html#:~:text=A%20random%20forest%20regressor.,accuracy%20and%20control%20over%2Dfitting. [1]

Cognizant

Image References

1) https://media.nature.com/lw800/magazine-assets/d41586-018-06619-3/d41586-018-06619-3_16101678.png
2) https://media.istockphoto.com/photos/excited-african-guy-showing-dollar-money-cash-over-blue-background-picture-id1342067848?b=1&k=20&m=1342067848&s=170667a&w=0&h=AemkCTK0p9FomQvnlGQBSsVV0mKulGnkZcW7gQLk2Is=
3) https://i.pinimg.com/originals/f3/05/8a/f3058a293fc7188f678a64ab174bae41.png
4) https://play-lh.googleusercontent.com/UrY7BAZ-XfXGpfkeWg0zCCeo-7ras4DCoRalC_WXXWTK9q5b0Iw7B0YQMsVxZaNB7DM
5) https://www.kaggle.com/static/images/logos/kaggle-logo-gray-300.png
6) https://encrypted-tbn0.gstatic.com/images?q=tbn:ANd9GcQ0IHPRIZI-UNI2g-K8CV4GSAMgdhOjGETT7g&usqp=CAU
7) https://blog.landr.com/wp-content/uploads/2020/06/Make-Money-from-Music_Feat_1200x627.jpg

Cognizant

Image References

- https://i.imgur.com/mtimFxh.png
- https://cdn.searchenginejournal.com/wp-content/uploads/2019/11/why-technical-seo-and-on-site-seo-is-rarely-enough-5dcfef7155db8.png
- https://3.bp.blogspot.com/-apoBeWFycKQ/XhKB8fEprwI/AAAAAAAACM4/SI76yzNSNYwlShIBrheDAum8L9qRtWNdgCLcBGAsYHQ/s1600/colab.png
- https://www.python.org/static/community_logos/python-logo-master-v3-TM.png
- https://miro.medium.com/max/1400/1*qgkjkj6BLVS1uD4mw_sTEg.png
- https://dwgeek.com/wp-content/uploads/2019/01/Spark-SQL-Performance-Tuning.jpg
- https://upload.wikimedia.org/wikipedia/commons/thumb/e/ed/Pandas_logo.svg/1200px-Pandas_logo.svg.png
- https://upload.wikimedia.org/wikipedia/commons/thumb/0/05/Scikit_learn_logo_small.svg/1200px-Scikit_learn_logo_small.svg.png

Cognizant

Image References

- https://user-images.githubusercontent.com/67586773/105040771-43887300-5a88-11eb-9f01-bee100b9ef22.png
- https://static.javatpoint.com/tutorial/matplotlib/images/matplotlib-tutorial.png
- https://miro.medium.com/max/412/1*Y-tB8ue9D1B7ZImpjYGXUA.png
- https://rapids.ai/assets/images/xgboost_logo.png
- https://mljar.com/images/machine-learning/random_forest_logo.png
- https://miro.medium.com/max/1400/1*chD302ssE0O62wreunGp4A.jpeg
- https://s3.amazonaws.com/ArchiveImages/LJ/2012/05/imagesCA1G7ZSW.jpg
- https://upload.wikimedia.org/wikipedia/commons/6/6f/Beethoven.jpg
- https://media1.popsugar-assets.com/files/thumbor/zan-t_Me63if8oqWYE9ENiPLIhA/0x224:2826x3050/fit-in/2048xorig/filters:format_auto-!!-:strip_icc-!!-/2020/02/11/894/n/1922398/87f6bb525e430e7bd44e40.22278576_/i/Drake.jpg