HOW TO COMPETE IN KAGGLE

A COMPLETE GUIDE TO KAGGLE COMPETITIONS

- MINCHANG (CARSON) ZHANG

TABLE OF CONTENT

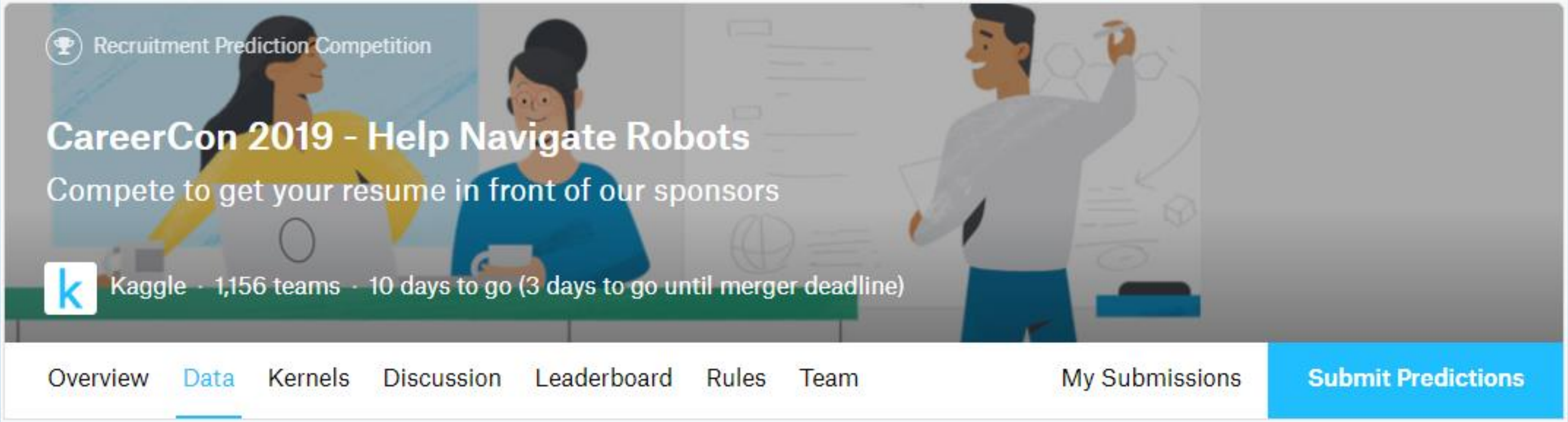
INTRODUCTION TO KAGGLE COMPETITION

- Kaggle is world's largest data scientist website
- Many companies launch competitions with a prize
- Kaggle competition provides "cleaned" and already split data
- Kaggle 2019 career con is coming, with a competition

Help Navigate Robots

Predict what surface the robot is on

HELP NAVIGATE ROBOTS



Data Sources

sample_submission.csv	3816 x 2
X_test.csv	488k x 13
X_train.csv	488k x 13
y_train.csv	3810 x 3

Columns

- series_id
- group_id
- surface

Columns

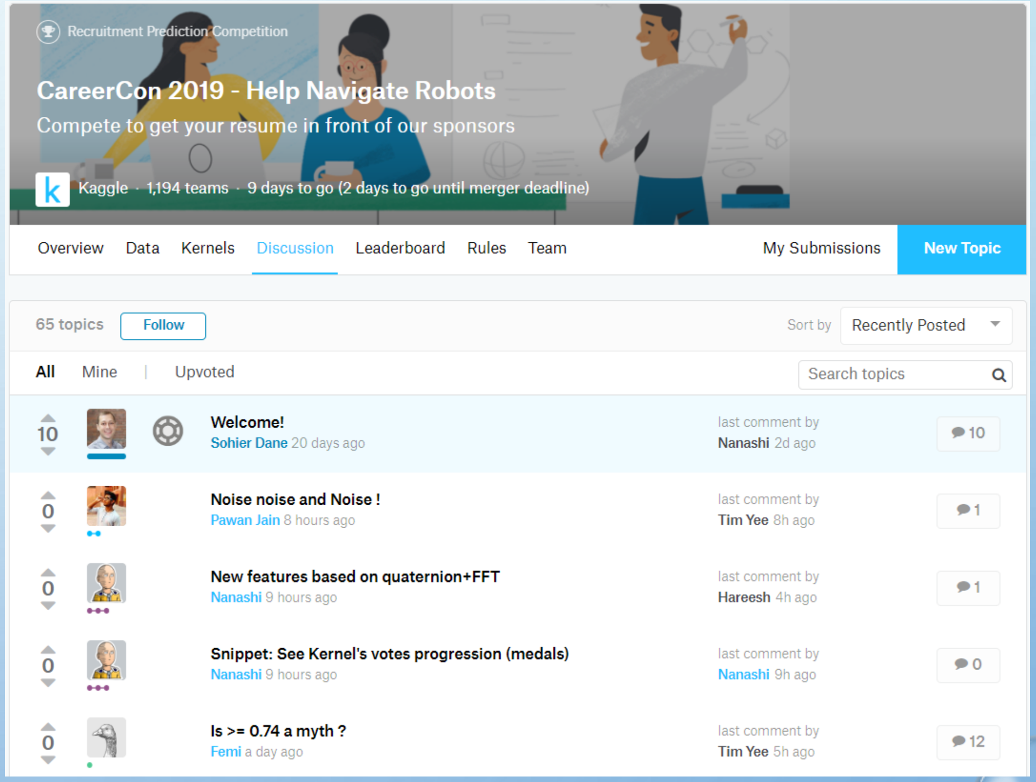
- row_id
- series_id
- measurement_number
- orientation_X
- orientation_Y
- orientation_Z
- orientation_W
- angular_velocity_X
- angular_velocity_Y
- angular_velocity_Z
- linear_acceleration_X
- linear_acceleration_Y
- linear_acceleration_Z

A GENERIC MACHINE LEARNING APPROACH

- Get the data
- Define problem
- Prepare data (data cleaning)
- Exploratory Data Analysis (EDA)
- Feature Engineering
- Modelling
- Evaluate Model Performance
- Hyperparameter Tuning
- Advanced Modelling and Optimization
- Repeat from feature selection
- Implementation and Periodically Model Training

KAGGLE COMPETITION APPROACH

- Get the data
- Define problem
- Prepare data (data cleaning)
- Exploratory Data Analysis (EDA)
- Feature Engineering
- Modelling
- Evaluate Model Performance
- Hyperparameter Tuning
- Advanced Modelling and Optimization
- Repeat from feature selection
- **Discussion**
- Implementation and Periodically Model Training



EDA – DF OPTIMIZATION

Dataframe optimization:

- ➢ Less ram used
- ➢ Faster in calculations

```
Memory usage of dataframe is 48.3692 MB
Memory usage after optimization is: 14.88 MB
Decreased by 69.2%
Memory usage of dataframe is 48.4454 MB
Memory usage after optimization is: 14.91 MB
Decreased by 69.2%
```

Ref: https://towardsdatascience.com/make-working-with-large-dataframes-easier-at-least-for-your-memory-6f52b5f4b5c4

EDA – TRAIN & TEST

```
This data has 487680 rows and 13 columns.

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 487680 entries, 0 to 487679
Data columns (total 13 columns):
row_id                   487680 non-null object
series_id                487680 non-null int16
measurement_number       487680 non-null int16
orientation_X            487680 non-null float16
orientation_Y            487680 non-null float16
orientation_Z            487680 non-null float16
orientation_W            487680 non-null float16
angular_velocity_X       487680 non-null float16
angular_velocity_Y       487680 non-null float16
angular_velocity_Z       487680 non-null float16
linear_acceleration_X    487680 non-null float16
linear_acceleration_Y    487680 non-null float16
linear_acceleration_Z    487680 non-null float16
dtypes: float16(10), int16(2), object(1)
memory usage: 14.9+ MB
```

	row_id	series_id	measurement_number	orientation_X	orientation_Y	orientation_Z	orientation_W	angular_velocity_X	angular_velo
0	0_0	0	0	-0.758301	-0.634277	-0.104858	-0.105957	0.107666	0.017563
1	0_1	0	1	-0.758301	-0.634277	-0.104919	-0.106018	0.067871	0.029938
2	0_2	0	2	-0.758301	-0.634277	-0.104919	-0.105957	0.007275	0.028931
3	0_3	0	3	-0.758301	-0.634277	-0.104980	-0.105957	-0.013054	0.019455
4	0_4	0	4	-0.758301	-0.634277	-0.104980	-0.105957	0.005135	0.007652

```
This data has 3810 rows and 3 columns.

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3810 entries, 0 to 3809
Data columns (total 3 columns):
series_id    3810 non-null int64
group_id     3810 non-null int64
surface      3810 non-null object
dtypes: int64(2), object(1)
memory usage: 89.4+ KB
```

	series_id	group_id	surface
0	0	13	fine_concrete
1	1	31	concrete
2	2	20	concrete
3	3	31	concrete
4	4	22	soft_tiles

EDA - TARGET

```
grouped = target.groupby('surface')
grouped.groups
```

```
{'carpet': Int64Index([  12,   13,   15,   16,   37,   54,   58,  118,  128,  149,
             ...
             3597, 3623, 3631, 3638, 3690, 3698, 3714, 3732, 3768, 3774],
            dtype='int64', length=189),
 'concrete': Int64Index([   1,    2,    3,    7,   14,   33,   34,   35,   48,   50,
             ...
             3765, 3767, 3769, 3770, 3778, 3780, 3787, 3791, 3795, 3798],
            dtype='int64', length=779),
 'fine_concrete': Int64Index([   0,   26,   32,   40,   42,   47,   56,   57,   75,   77,
             ...
             3760, 3763, 3766, 3772, 3782, 3783, 3786, 3797, 3800, 3807],
            dtype='int64', length=363),
 'hard_tiles': Int64Index([  27,   45,  148,  189,  257,  459,  527,  566,  587,  745,  798,
              804,  826, 1125, 1193, 1277, 1399, 1454, 1455, 1610, 1671],
            dtype='int64'),
 'hard_tiles_large_space': Int64Index([   8,   21,   29,   63,   98,  119,  124,  142,  153,  1
65,
```

```
grouped.get_group('hard_tiles')
```

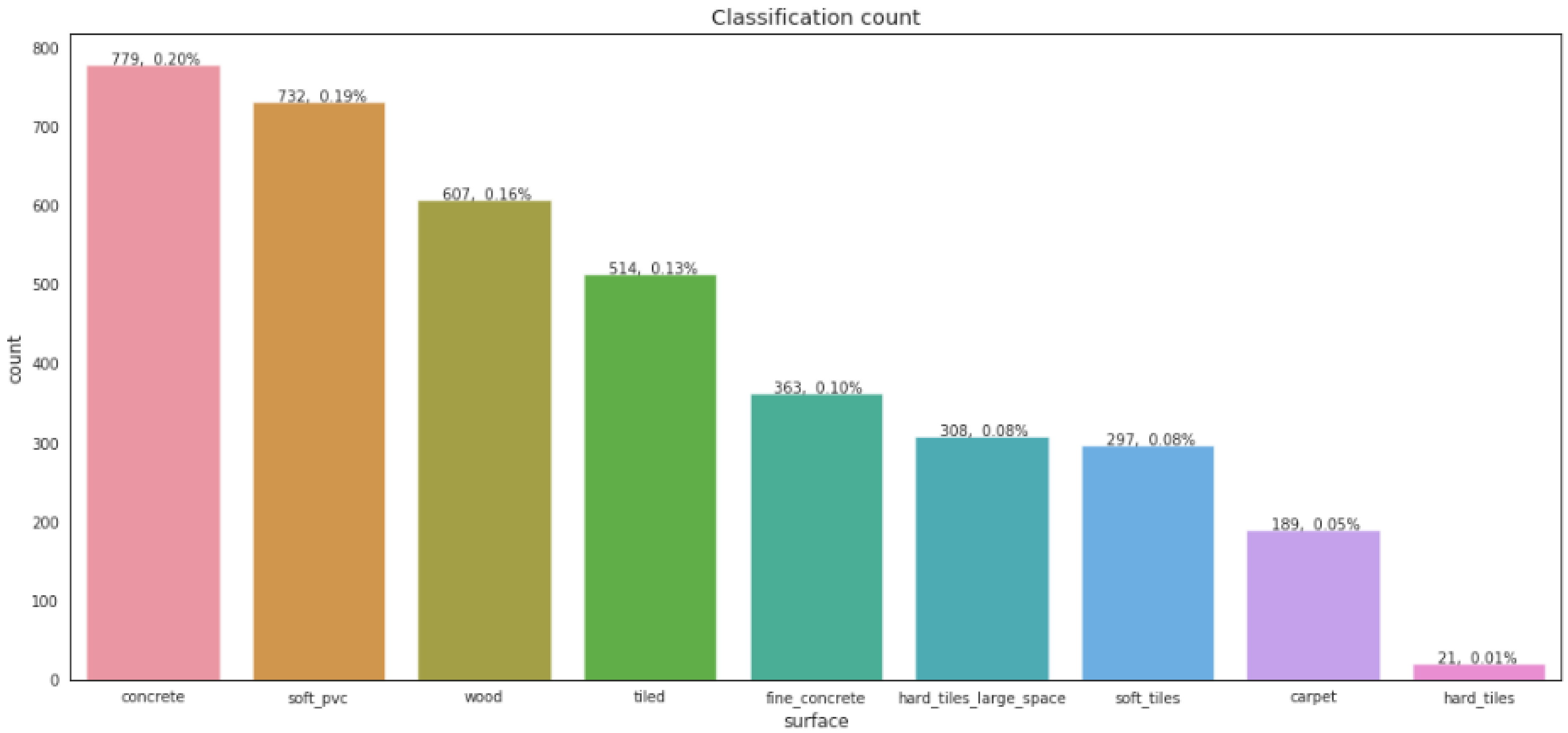
	series_id	group_id	surface
27	27	27	hard_tiles
45	45	27	hard_tiles
148	148	27	hard_tiles
189	189	27	hard_tiles
257	257	27	hard_tiles
459	459	27	hard_tiles
527	527	27	hard_tiles
566	566	27	hard_tiles
587	587	27	hard_tiles
745	745	27	hard_tiles
798	798	27	hard_tiles
804	804	27	hard_tiles
826	826	27	hard_tiles
1125	1125	27	hard_tiles
1193	1193	27	hard_tiles
1277	1277	27	hard_tiles
1399	1399	27	hard_tiles
1454	1454	27	hard_tiles
1455	1455	27	hard_tiles
1610	1610	27	hard_tiles
1671	1671	27	hard_tiles

```
grouped.get_group('concrete')['group_id'].unique()
```

```
array([31, 20, 12, 32,  0,  5, 62, 41, 42, 61, 57, 47, 39, 50, 63])
```

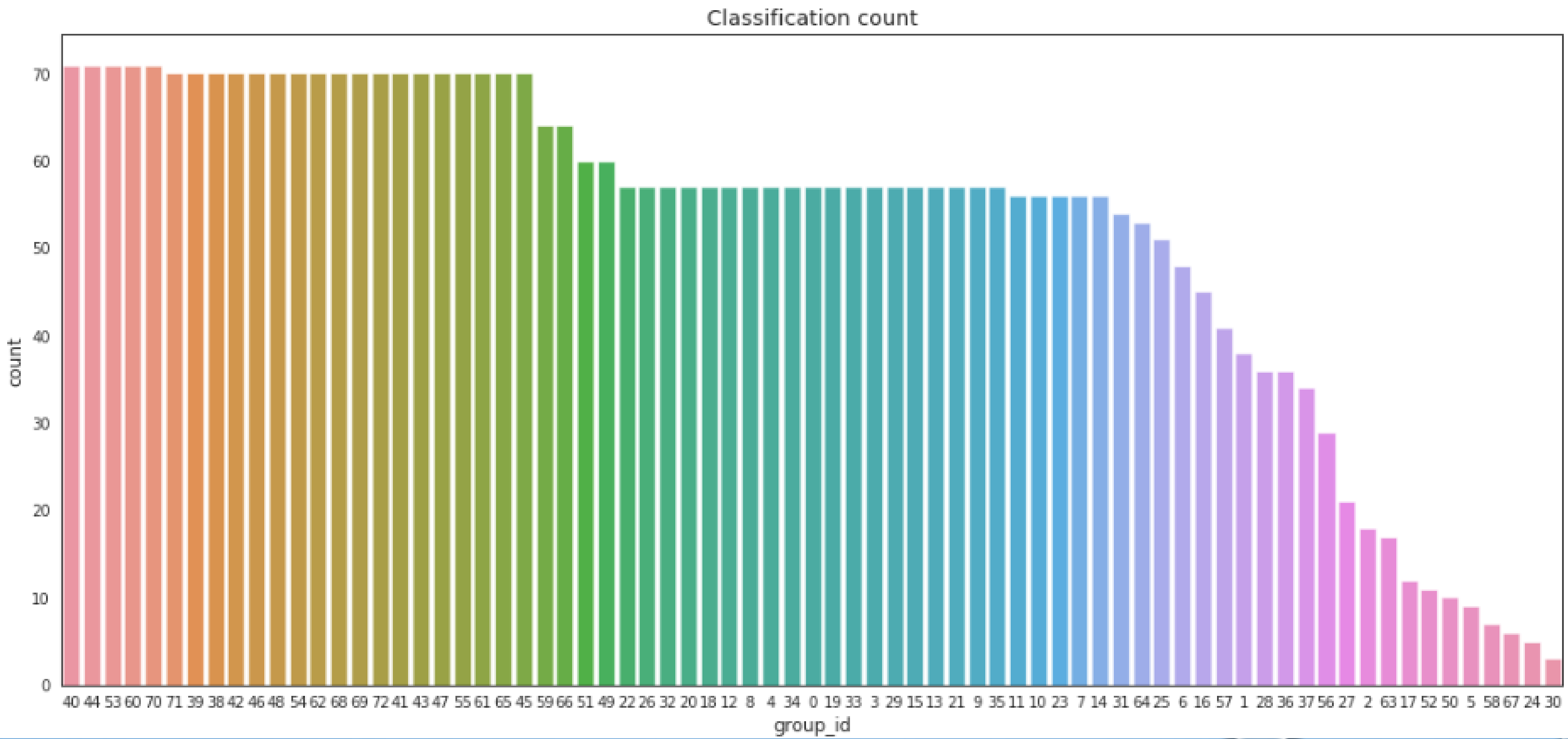
EDA – TARGET PLOT

```
plot_count(target['surface'])
```

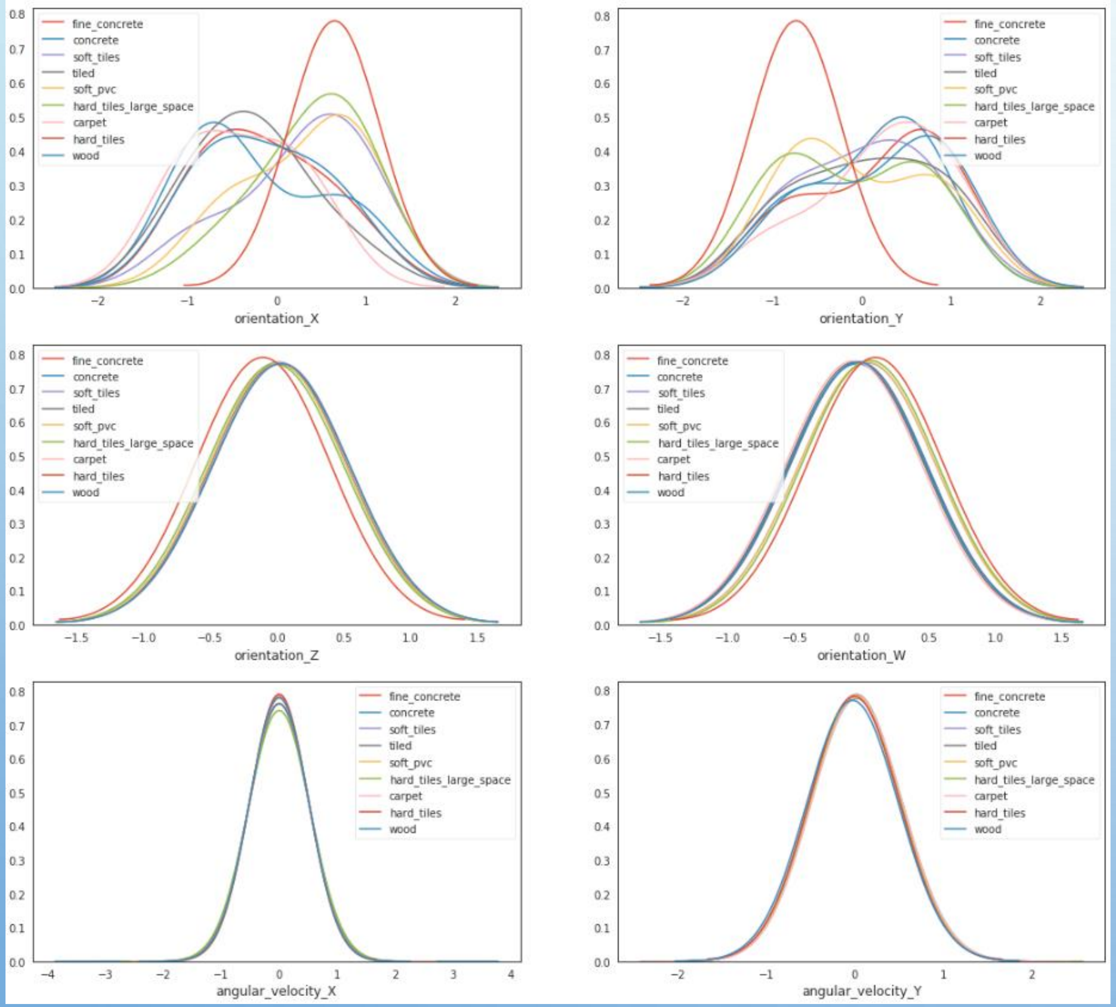


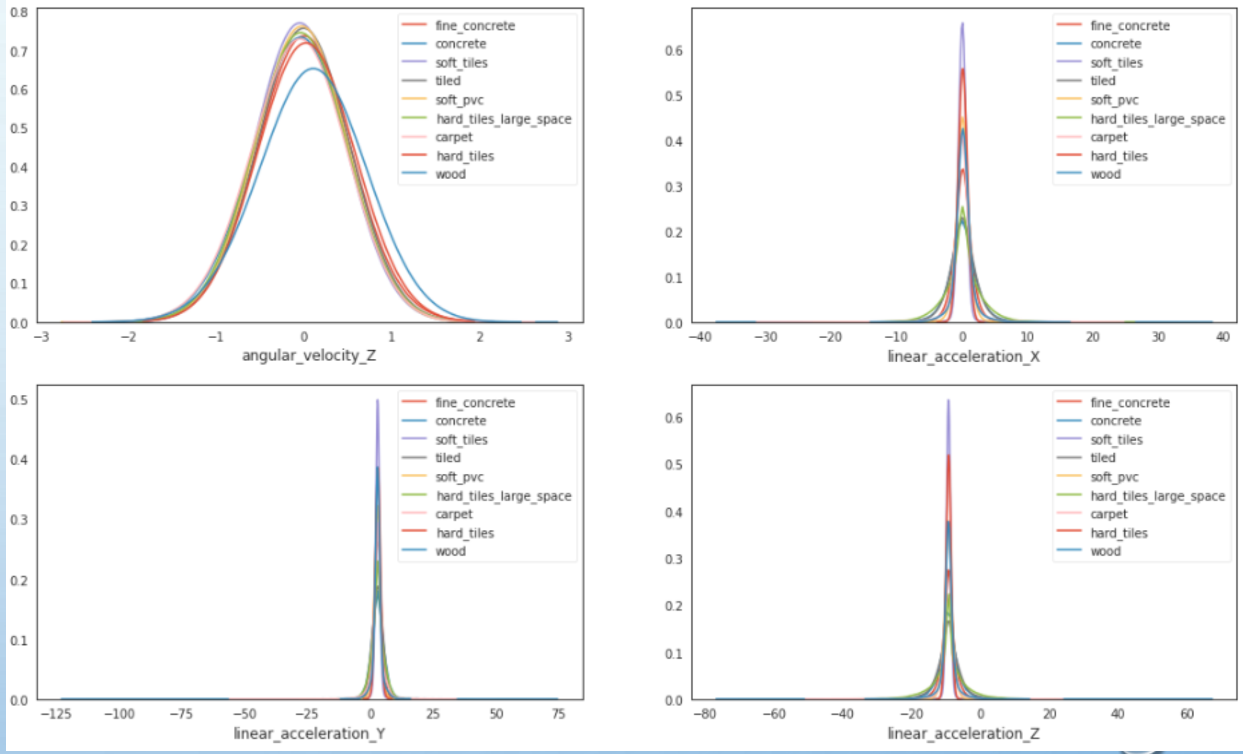
EDA – TARGET GROUPID

```
plot_count(target['group_id'], annotation=False)
```



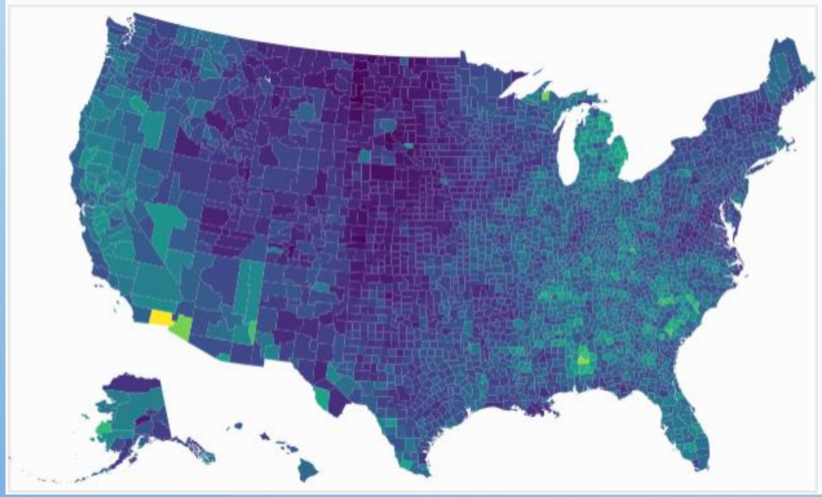
EDA – TARGET TIME SERIES PLOT

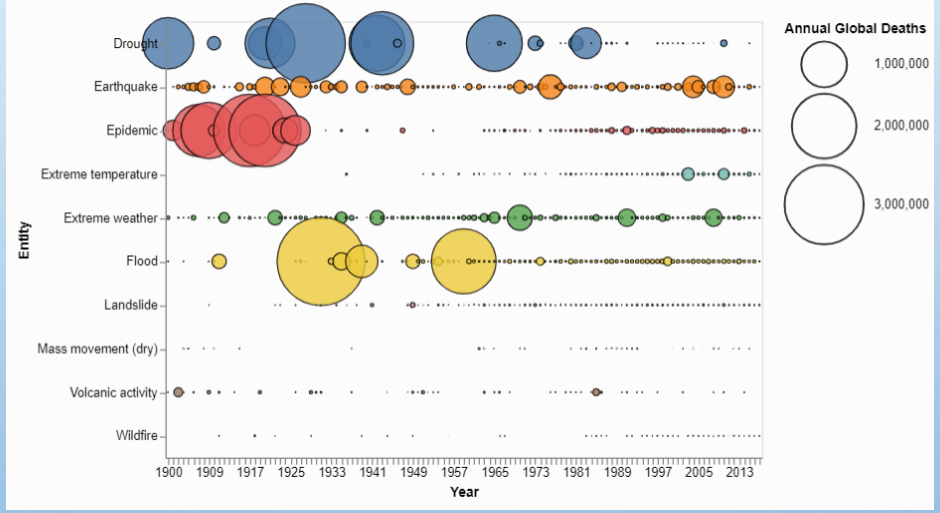


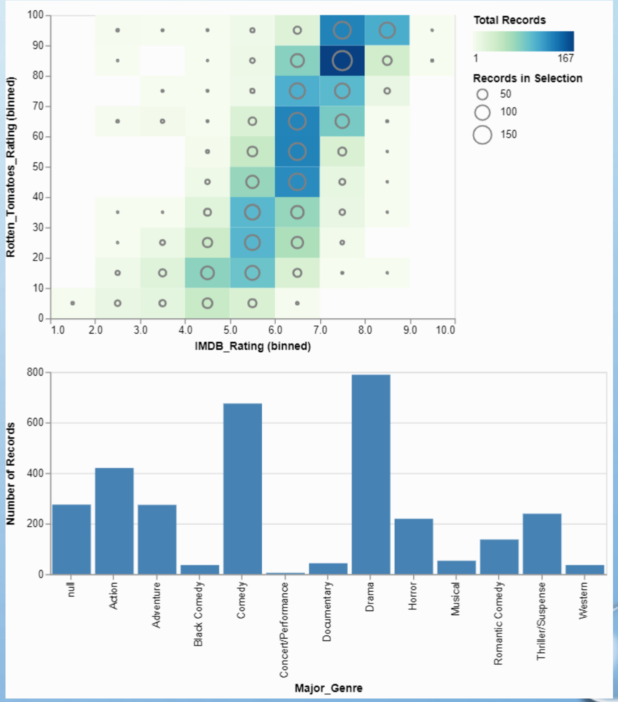


OTHER VISUALIZATION PACKAGES

- Other than matplotlib and seaborn, there are a couple other packages
- Plotly is really nice, but it COST
- Altair is really cool, allows custom html render







Ref: https://www.kaggle.com/notslush/altair-visualization-2018-stackoverflow-survey
https://altair-viz.github.io/

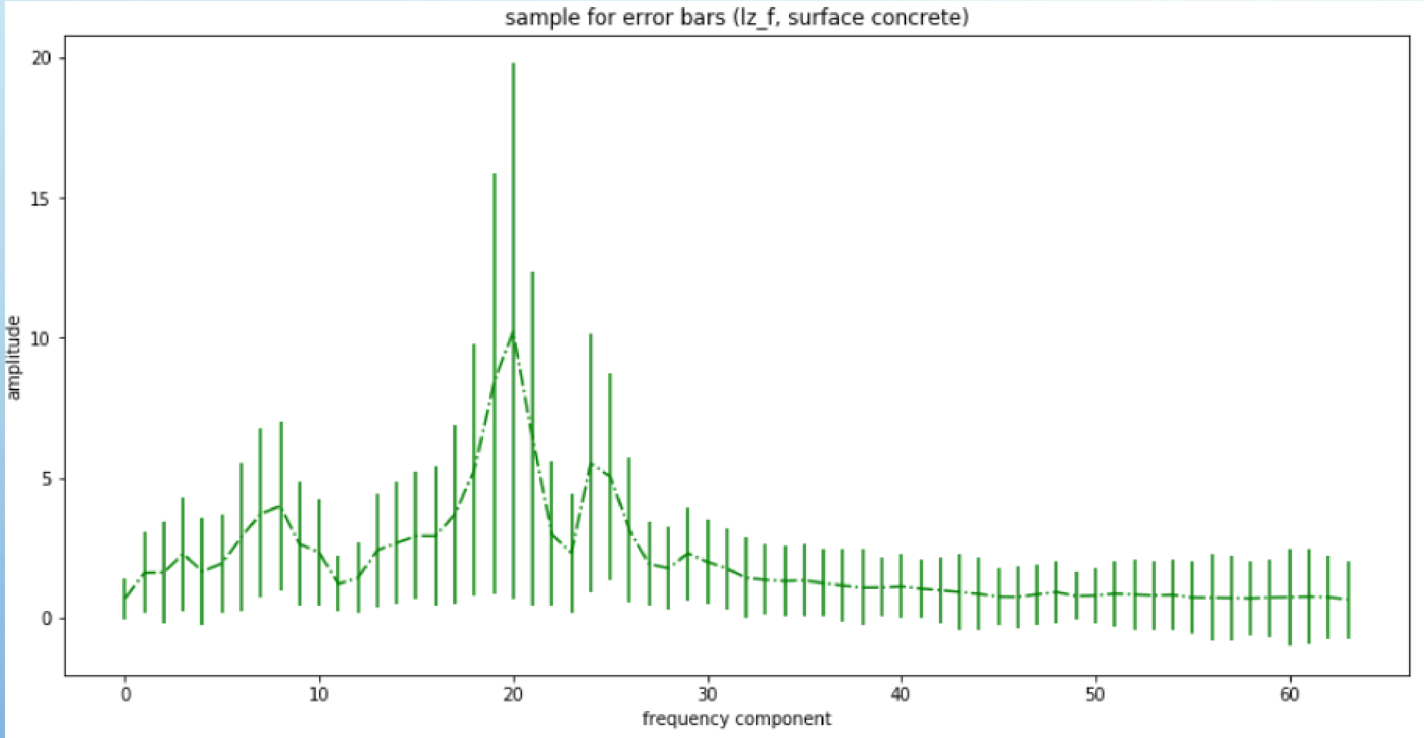
FEATURE ENGINEERING

- Statistical data: mean, max, min, std, variance
- Physics data: total, acceleration, range, absolute, change rate
- Signal processing: Fourier transform, frequency domain analysis

MODELLING

- First approach: merge train and target data, predict directly, choose the max number of predicted surface for each series as result
  - CV: 0.6, LB: 0.34
- Second approach: group data based on series id
  - CV: 0.82, LB: 0.4
- Third approach: grouped data with cross validation
  - CV: 0.88, LB: 0.62
- Fourth approach: grouped data with cross validation and Beysian optimization
  - CV: 0.92, LB: 0.62
- Fifth approach: grouped data with cross validation (StratifiedKFold) and Beysian optimization
  - CV: 0.91, LB: 0.64
- Sixth approach: ensemble with max voting with cross validation (StratifiedKFold)
  - CV: 0.88, LB: 0.66
- Seventh approach: ensemble with stacking with cross validation (StratifiedKFold)
  - CV: 0.86, LB: 0.63

REVISIT OPTIMIZATION STRATEGY



- Try again with custom train test split function
- Make sure not to share a group between training and validation
- No cross validation due to time constraint
- CV: 0.56, LB: 0.54
- Unrealistic CV score was brought down!
- I'm on the right track!

Ref: https://www.kaggle.com/trohwer64/simple-fourier-analysis

WHAT TO DO NEXT

- Better data split and cross validation strategy
- Try Neural Network
- Improve ensemble
- Participate in discussion

SUMMARY

- Machine learning approach
- Discussion can be super useful in Kaggle competitions
- Should you trust CV score or LB score?
- Cross validation is key