NANYANG TECHNOLOGICAL UNIVERSITY SINGAPORE

Modelling Earthquake Damage

CZ1015 Mini Project:

Li Haoyang, Png Yao Wei Samuel, Tng Jun Wei, Wei Kaitao

Outline

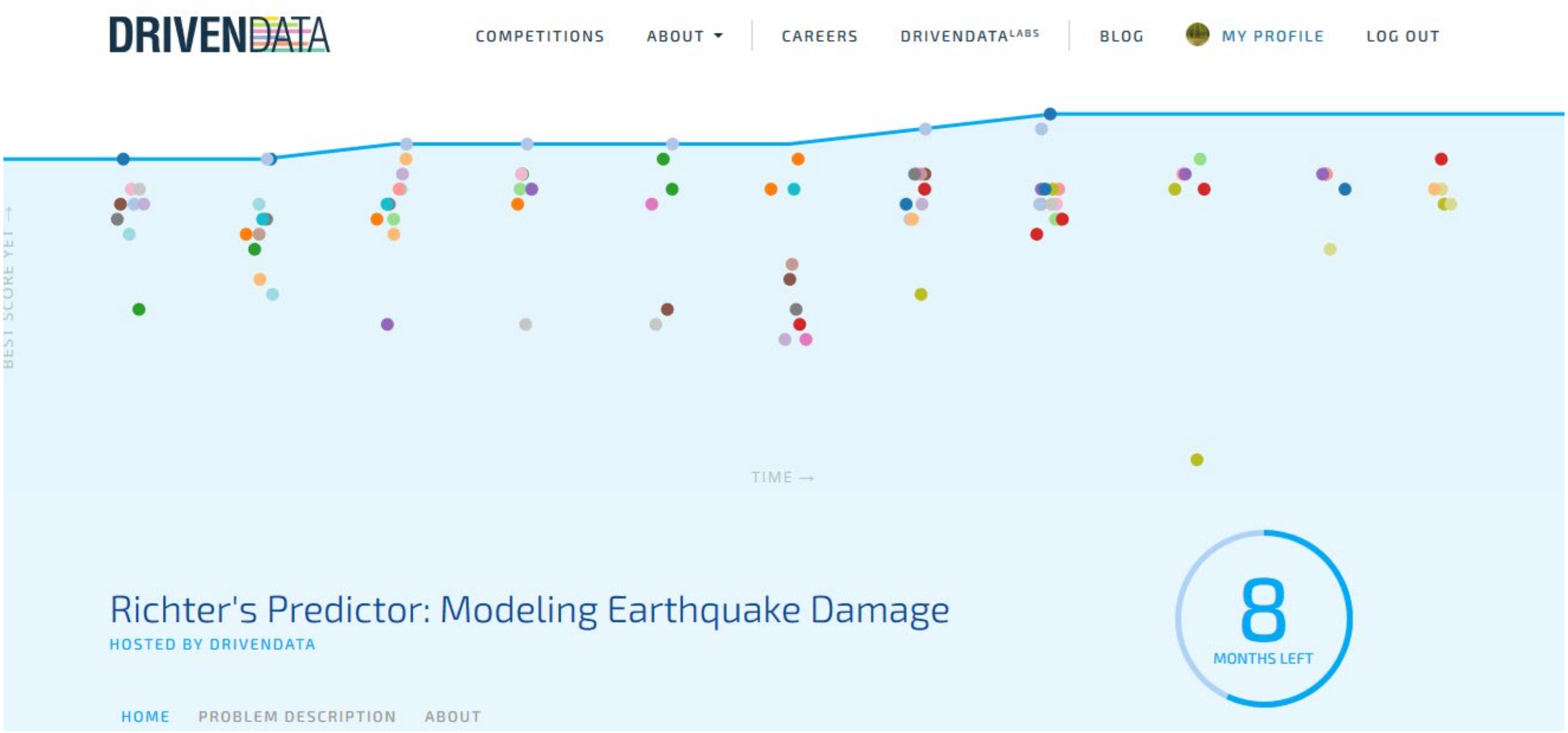
Part I. Problem Description

- Major earthquake in Gorkha District, Nepal
- Magnitude **7.8**
- **>9000** casualties
- Est. US$6b losses, **35%** of Nepal GDP

Source: https://www.britannica.com/topic/Nepal-earthquake-of-2015

We aim to develop a **multiclass-classification** model to predict the potential severity of damage on each building

Part II. Data Acquisition



Source: https://www.drivendata.org/competitions/57/nepal-earthquake/page/134/

Part III. Data Exploration

260601 train data (labels and features)

86868 test data (features only)

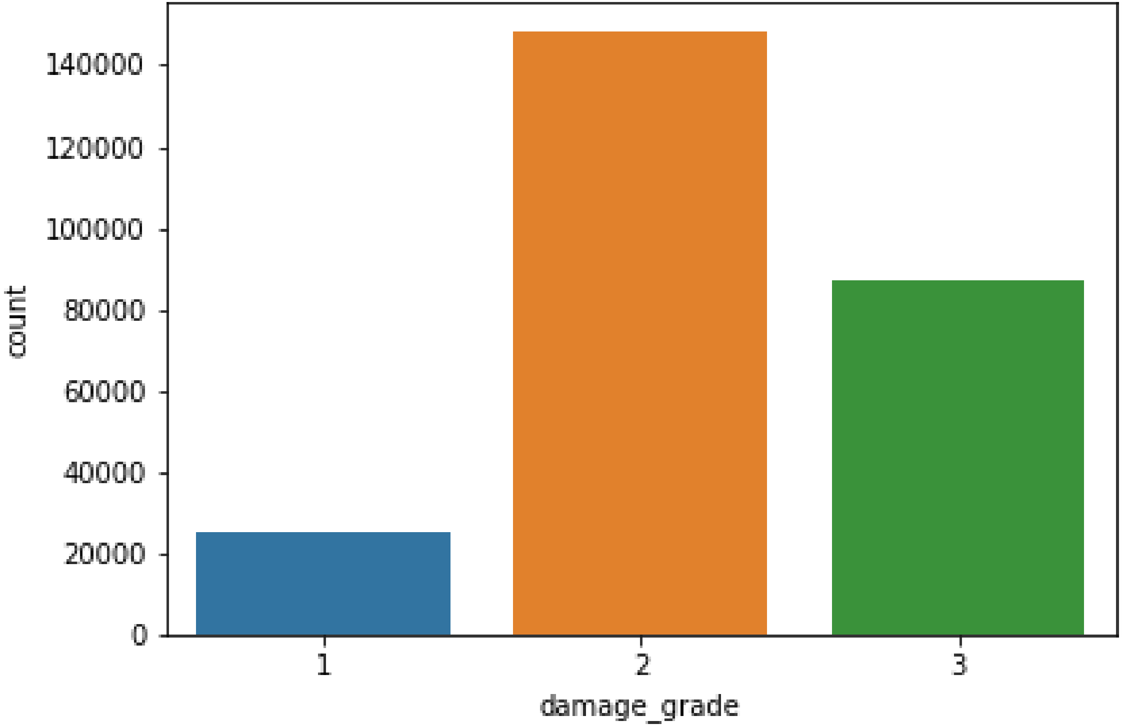
Total features: 38

- Numeric: 5
- Categorical (binary): 22
- Categorical (multi-nary): 11

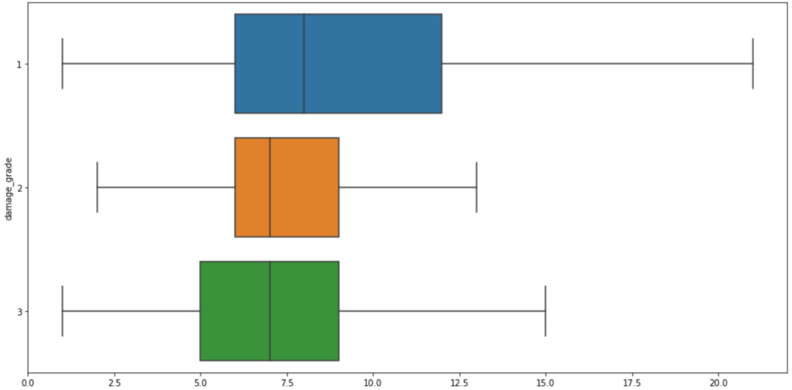
- `geo_level_1_id`, `geo_level_2_id`, `geo_level_3_id` (type: int): geographic region in which building exists, from largest (level 1) to most specific sub-region (level 3). Possible values: level 1: 0-30, level 2: 0-1427, level 3: 0-12567.
- `count_floors_pre_eq` (type: int): number of floors in the building before the earthquake.
- `age` (type: int): age of the building in years.
- `area_percentage` (type: int): normalized area of the building footprint.
- `height_percentage` (type: int): normalized height of the building footprint.
- `land_surface_condition` (type: categorical): surface condition of the land where the building was built. Possible values: n, o, t.
- `foundation_type` (type: categorical): type of foundation used while building. Possible values: h, i, r, u, w.
- `roof_type` (type: categorical): type of roof used while building. Possible values: n, q, x.
- `ground_floor_type` (type: categorical): type of the ground floor. Possible values: f, m, v, x, z.
- `other_floor_type` (type: categorical): type of constructions used in higher than the ground floors (except of roof). Possible values: j, q, s, x.
- `position` (type: categorical): position of the building. Possible values: j, o, s, t.
- `plan_configuration` (type: categorical): building plan configuration. Possible values: a, c, d, f, m, n, o, q, s, u.
- `has_superstructure_adobe_mud` (type: binary): flag variable that indicates if the superstructure was made of Adobe/Mud.
- `has_superstructure_mud_mortar_stone` (type: binary): flag variable that indicates if the superstructure was made of Mud Mortar - Stone.
- `has_superstructure_stone_flag` (type: binary): flag variable that indicates if the superstructure was made of Stone.
- `has_superstructure_cement_mortar_stone` (type: binary): flag variable that indicates if the superstructure was made of Cement Mortar - Stone.
- `has_superstructure_mud_mortar_brick` (type: binary): flag variable that indicates if the superstructure was made of Mud Mortar - Brick.
- `has_superstructure_cement_mortar_brick` (type: binary): flag variable that indicates if the superstructure was made of Cement Mortar - Brick.
- `has_superstructure_timber` (type: binary): flag variable that indicates if the superstructure was made of Timber.
- `has_superstructure_bamboo` (type: binary): flag variable that indicates if the superstructure was made of Bamboo.
- `has_superstructure_rc_non_engineered` (type: binary): flag variable that indicates if the superstructure was made of non-engineered reinforced concrete.
- `has_superstructure_rc_engineered` (type: binary): flag variable that indicates if the superstructure was made of engineered reinforced concrete.
- `has_superstructure_other` (type: binary): flag variable that indicates if the superstructure was made of any other material.
- `legal_ownership_status` (type: categorical): legal ownership status of the land where building was built. Possible values: a, r, v, w.
- `count_families` (type: int): number of families that live in the building.
- `has_secondary_use` (type: binary): flag variable that indicates if the building was used for any secondary purpose.
- `has_secondary_use_agriculture` (type: binary): flag variable that indicates if the building was used for agricultural purposes.
- `has_secondary_use_hotel` (type: binary): flag variable that indicates if the building was used as a hotel.
- `has_secondary_use_rental` (type: binary): flag variable that indicates if the building was used for rental purposes.
- `has_secondary_use_institution` (type: binary): flag variable that indicates if the building was used as a location of any institution.
- `has_secondary_use_school` (type: binary): flag variable that indicates if the building was used as a school.
- `has_secondary_use_industry` (type: binary): flag variable that indicates if the building was used for industrial purposes.
- `has_secondary_use_health_post` (type: binary): flag variable that indicates if the building was used as a health post.
- `has_secondary_use_gov_office` (type: binary): flag variable that indicates if the building was used fas a government office.
- `has_secondary_use_use_police` (type: binary): flag variable that indicates if the building was used as a police station.
- `has_secondary_use_other` (type: binary): flag variable that indicates if the building was secondarily used for other purposes.

Source: DrivenData: https://www.drivendata.org/competitions/57/nepal-earthquake/page/136/

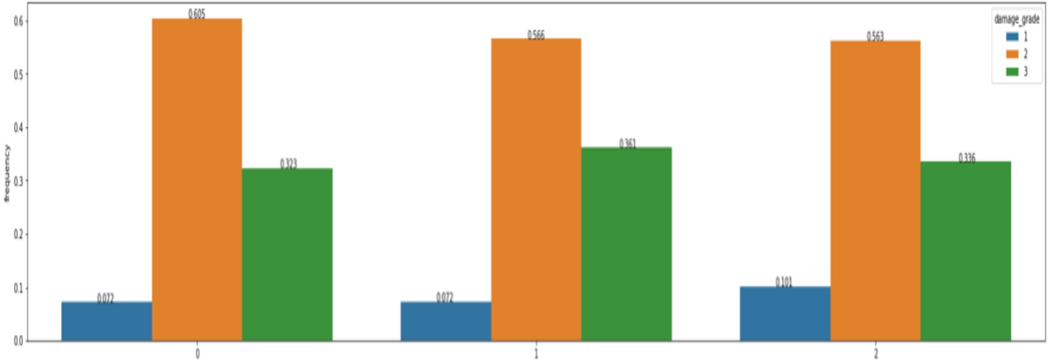
Univariate analysis on damage_grade



Bivariate analysis with damage_grade



area_percentage



land_surface_condition

Most predictors show some relation.
e.g. area of house

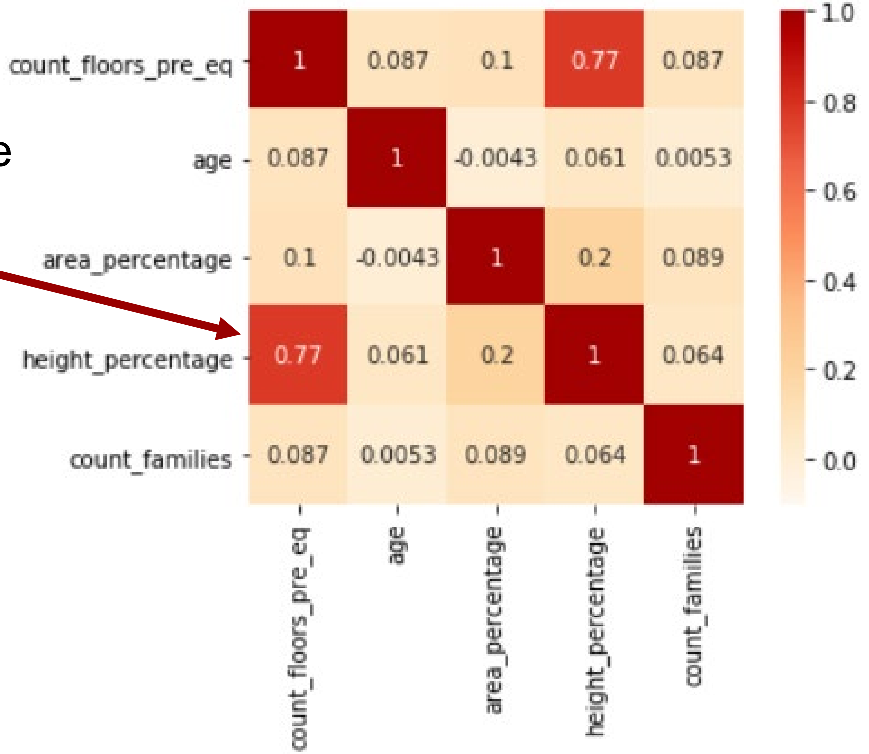
Some predictors show little relation
e.g. land surface condition

Part III. Data Exploration

Bivariate analysis between predictors

Discover:
The count_floor_per_eq and height_percentage have high correlation (0.77).

Decision:
Remove one of them, we choose count_floor_pre_eq



Part IV. Data Pre-processing

Combined train data and test data

⇩

Dropped some features

⇩

Normalized numeric features

⇩

One-hot encode categorical features

Part IV. Data Pre-processing

Separate dataset

⇩

Splitted data into train data and validation data

⇩

Oversample a copy of the new train data

Part V & VI. Data Analysis & Results Analysis

I. Logistic Regression

II. Neural Network

III. Random Forest

IV. Support Vector Machine

Overall Approach for Model Selection

1. Find out if oversampling, removal of less important features improve models' performance

1. Train a basic model using the train set as baseline

1. Optimise model through hyperparameter selection

3. Apply the models on actual test data (submit for competition)

3. Select best model based on F1 Micro

Part V & VI. Data Analysis & Results Analysis

Investigation: Oversampling

Model	Validation F1-micro without oversampling	Validation F1-micro with oversampling	Change in F1-micro
Logistic Regression	0.667	0.581	0.086
SVM	0.643	0.557	0.086
MLP	0.677	0.599	0.078
Random Forest	0.656	0.617	0.039

Part V & VI. Data Analysis & Results Analysis

Decision: Did not oversample the datasets

Investigation: Feature Removal

Model	Validation F1-micro without removing features	Validation F1-micro after removing features	Change in F1-micro
Logistic Regression	0.676	0.667	↓ 0.009
SVM	0.647	0.643	↓ 0.004
MLP	0.681	0.677	↓ 0.004
Random Forest	0.670	0.656	↓ 0.014

Part V & VI. Data Analysis & Results Analysis

Decision:
Did not remove the features
Except SVM

Logistic regression

Hyperparameter tuning

- Used random Search to optimize the regularization parameter
- Negligible change in performance

	Before random search	After random search
Validation f1-micro	0.67583	0.67581 ↓
Test f1-micro	0.6711	0.6711

Logistic regression

	Train	Validation
F1-micro before random search	0.669	0.676

differs by only 0.007

Why is hyperparameter tuning ineffective?

- Negligible overfitting issue

How to improve?

- Add more features

Neural Network

- Basic multi-layer perceptron network classifier, provided by *sklearn.neural_network*

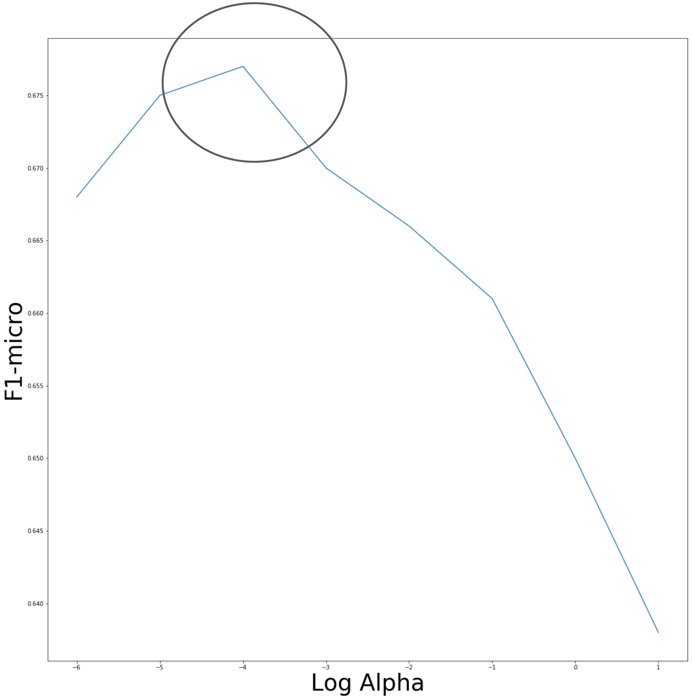
⇩

- Hyperparameter Optimisation
  - Two parameters: alpha, hidden layers
  - Performed GridSearch on two sets of specified hyperparameters

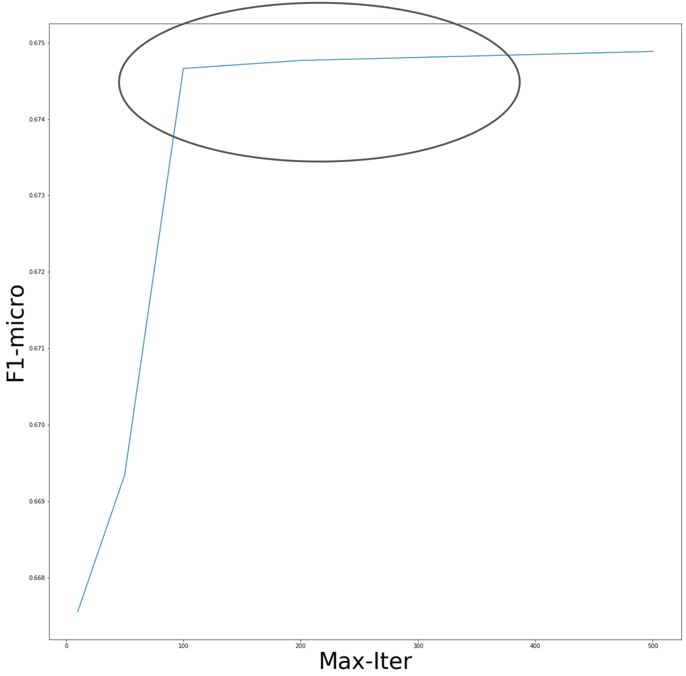
⇩

- Parameter optimisation
  - Set number of iterations to 5000

Part V & VI. Data Analysis & Results Analysis



F1-micro of val. vs Log of alpha



F1-micro of val. vs No. of iterations

Part V & VI. Data Analysis & Results Analysis

Model	F1-micro
Validation	0.681
Test	0.689

Random Forest

Hyperparameter optimization

- 10-fold Cross-Validation
- Scoring based on F1 Micro

Results Analysis

- Slight increase in F1 Score
- F1 score more consistent between seen and unseen data

Key Parameters

	Default	Random Search	Optimized Results
Number of trees	100	10 to 200	44
Maximum depth	None	None, 3 to 20	11
Maximum features per split	√Features	√, Log, All, 0.5 to 0.9	0.8 * Features
F1 Micro Train	0.92619	-	0.67080 ↓
F1 Micro Validation	0.67042	-	0.67082 ↑

Random Forest

Model Findings

- Cross validation lowered importance for less important predictors and increased importance for more important predictors
- Reduced overfitting
- More accurate model representation

How to improve optimization?

- Increase the range of trees to 1000 (Will require a lot of memory and time)
- Test for deeper trees to reduce bias

Key Features Importance

Feature	Importance	Importance (Optimized)
Number of families	0.033649	0.003029 ↓
Foundation Type R	0.011989	0.146175 ↑
Ground Floor Type V	0.009841	0.051839 ↑

Support Vector Machine (SVM)

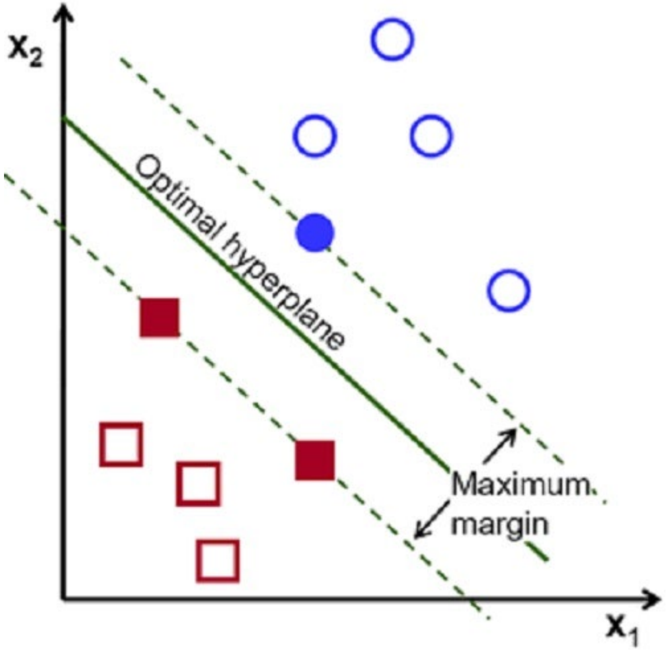
Optimization:

- Data preprocess.
  - Drop less relative data

- Tweak hyperparameters
  - Iteration number.
  - C-value: Punishment on wrong responses.
  - Class weight: uniform/balanced.
  - ….

After first two optimizations:

F1 Score: ~0.645



Source: https://www.aitrends.com/ai-insider/support-vector-machines-svm-ai-self-driving-cars/

Support Vector Machine (SVM)

Kernel Trick Optimization:

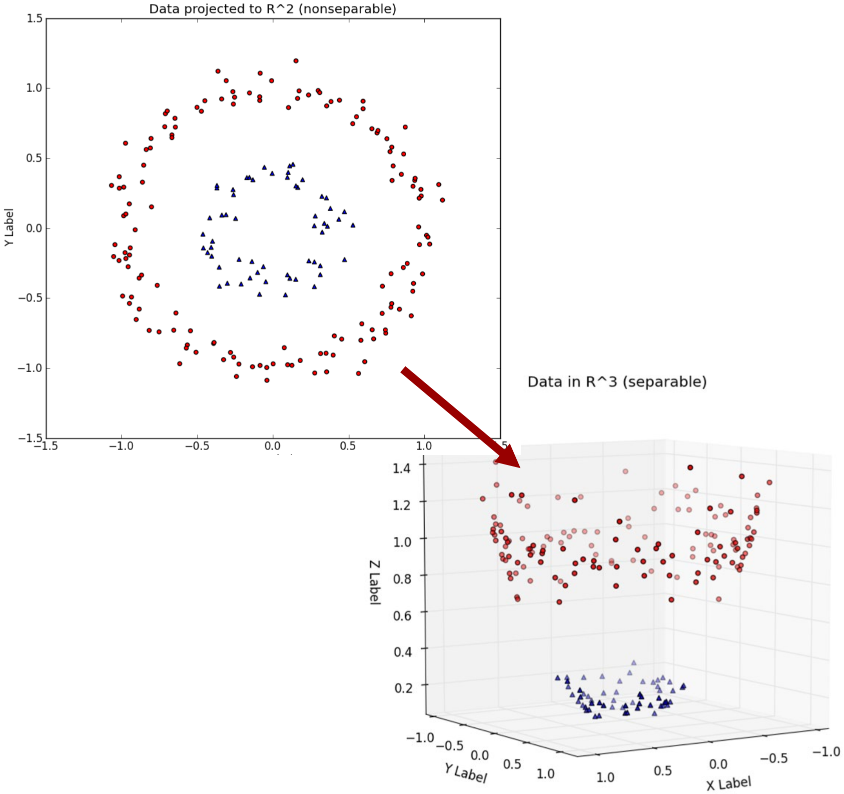
- Polynomial function (3 degree)
- Radial basis function (RBF)
- Sigmoid function

Without Kernel Trick:

F1 Score: ~0.645

With RBF Kernel:

F1 Score: ~0.667 (↑ 0.022)



Comparing the competition scores for different algorithms

Ranking	Optimized Model	Competition score
1	Neural Network	0.689
2	Support Vector Machine	0.677
3	Logistic regression	0.671
4	Random Forest	0.670

Part VII. Conclusion

We met our objective!

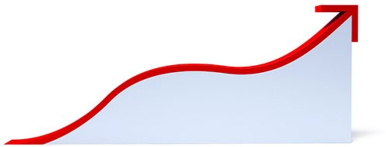
We successfully developed several models with decent performance

- Our best model: 0.6890
- We beat 79.1% of the total 2400+ competitors

Part VII. Conclusion

Things we learned

- Techniques we didn’t learn in course
- Bias and oversampling
- Dropping features is not simple
- Different model need different strategies



Improvements

- Upgrade hardware
- Use geo_level_2_id, geo_level_3_id
- Use PCA to reduce dimensionality of correlated features
- Use other frameworks to exploit parallelism.

………..

Contributions

Samuel
- Random Forest
- Bivariate exploration
- Data preprocessing
  - Data encoding
- Presentation slides

Haoyang
- Logistic Regression
- Bivariate exploration
- Data preprocessing
  - The rest
- Presentation slides

Junwei
- MLP
- Univariate exploration
- Data preprocessing
  - Oversampling
  - The rest
- Presentation slides

Kaitao
- SVM
- Univariate exploration
- Data Preprocessing
  - Data encoding
  - Oversampling
- Presentation slides