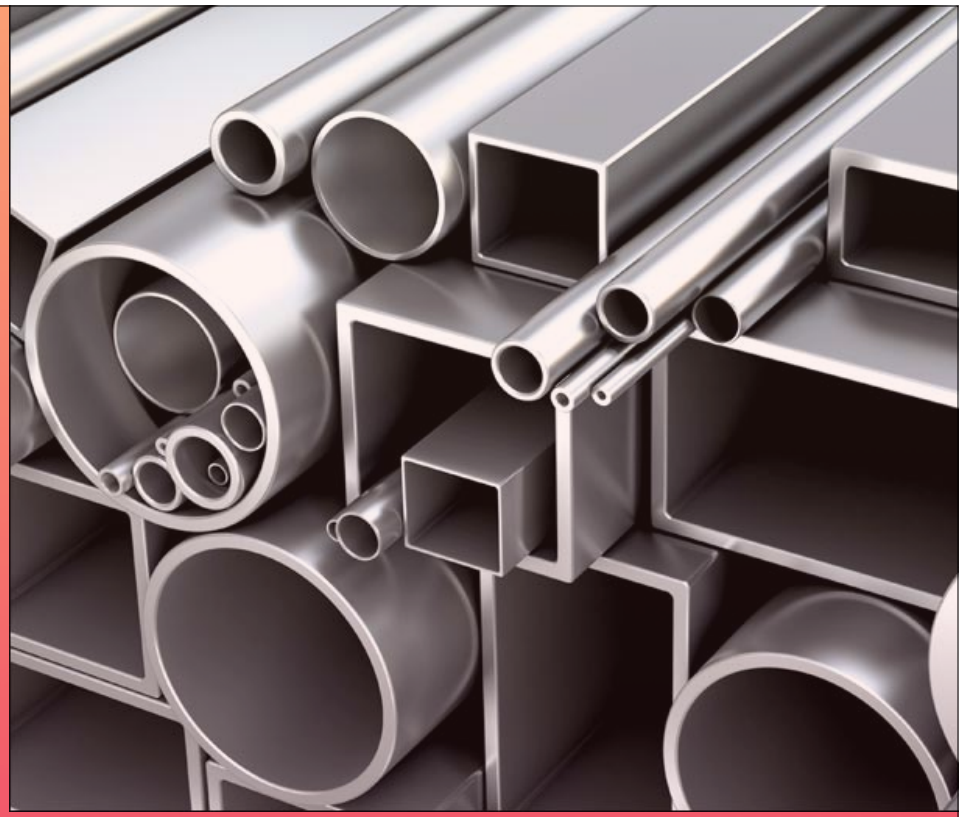
Steel Grade Classification

James Fan I Metis SF I 8/7/19



Automotive Parts

Kitchen Utensils

Disney Concert Hall

Companies over specify more expensive materials, when more cost efficient alternatives are available.

Steel Specification

2.1 Chemical Composition of Carbon Steels for General Use (Continued)

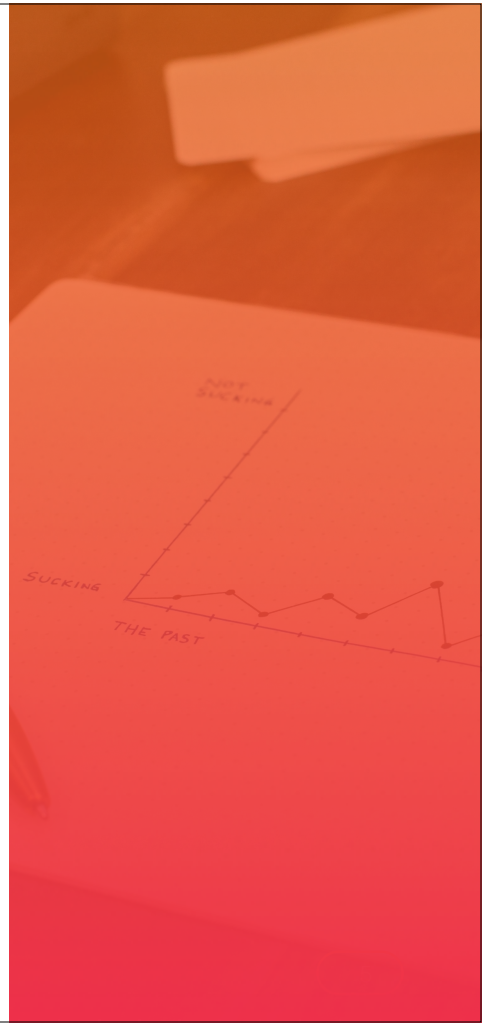
Standard Designation	Grade, Class, Type, Symbol or Name	Steel Number	UNS Number	Weight, %, max, Unless Otherwise Specified								
				C	Mn	Si	P	S	Cr	Ni	Mo	Others
ASTM A 29/A 29M-03	1049	---	G10490	0.46-0.53	0.60-0.90	---	0.040	0.050	---	---	---	---
	1050	---	G10500	0.48-0.55	0.60-0.90	---	0.040	0.050	---	---	---	---
	1053	---	G10530	0.48-0.55	0.70-1.00	---	0.040	0.050	---	---	---	---
ASTM A 108-03	1050	---	G10500	0.48-0.55	0.60-0.90	---	0.040	0.050	---	---	---	---
ASTM A 576-90b (2000)	1049	---	G10490	0.46-0.53	0.60-0.90	---	0.040	0.050	---	---	---	---
	1050	---	G10500	0.48-0.55	0.60-0.90	---	0.040	0.050	---	---	---	---
	1053	---	G10530	0.48-0.55	0.70-1.00	---	0.040	0.050	---	---	---	---
SAE J403 NOV01	1049	---	G10490	0.46-0.53	0.60-0.90	---	0.030	0.050	---	---	---	---
	1050	---	G10500	0.48-0.55	0.60-0.90	---	0.030	0.050	---	---	---	---
	1053	---	G10530	0.48-0.55	0.70-1.00	---	0.030	0.050	---	---	---	---
JIS G 4051:1979	S 50 C	---	---	0.47-0.53	0.60-0.90	0.15-0.35	0.030	0.035	0.20	0.20	---	Cu 0.30; Ni+Cr 0.35
EN 10016-2:1995	C50D	1.0586	---	0.48-0.53	0.50-0.80	0.10-0.30	0.035	0.035	0.15	0.20	0.05	Cu 0.25; Al 0.01
EN 10016-4:1995	C50D2	1.1171	---	0.48-0.52	0.50-0.70	0.10-0.30	0.020	0.025	0.10	0.10	0.03	Cu 0.15; Al 0.01; N 0.007
EN 10083-1:1991	C50E	1.1206	---	0.47-0.55	0.60-0.90	0.40	0.035	0.035	0.40	0.40	0.10	Cr+Mo+Ni 0.63
	C50R	1.1241	---	0.47-0.55	0.60-0.90	0.40	0.035	0.020-0.040	0.40	0.40	0.10	Cr+Mo+Ni 0.63
EN 10083-2:1991	C50	1.0540	---	0.47-0.55	0.60-0.90	0.40	0.045	0.045	0.40	0.40	0.10	Cr+Mo+Ni 0.63
ISO 683-1:1987	C 50	---	---	0.47-0.55	0.60-0.90	0.10-0.40	0.045	0.045	---	---	---	---
	C 50 E4	---	---	0.47-0.55	0.60-0.90	0.10-0.40	0.035	0.035	---	---	---	---
	C 50 M2	---	---	0.47-0.55	0.60-0.90	0.10-0.40	0.035	0.020-0.040	---	---	---	---
JIS G 4051:1979	S 53 C	---	---	0.50-0.56	0.60-0.90	0.15-0.35	0.030	0.035	0.20	0.20	---	Cu 0.30; Ni+Cr 0.35
EN 10016-2:1995	C52D	1.0588	---	0.50-0.55	0.50-0.80	0.10-0.30	0.035	0.035	0.15	0.20	0.05	Cu 0.25; Al 0.01
EN 10016-4:1995	C52D2	1.1202	---	0.50-0.54	0.50-0.70	0.10-0.30	0.020	0.025	0.10	0.10	0.03	Cu 0.15; Al 0.01; N 0.007

Source: Handbook of Comparable World Steel Standards 3rd Edition John E Bringas

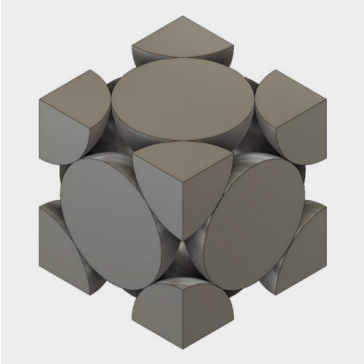
Problem Statement

- There are thousands of different steel specifications
- Countries have their own standard with different names
- So many options, choosing the right one is hard

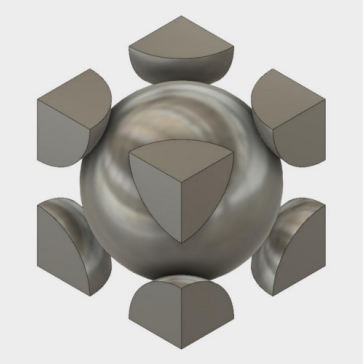
How do you choose the one that works?



Steel Type Characteristics



Face Centered Cubic



Body Centered Cubic

Allotrope	Corrosion Resistance	Hardness	Crystal Structure	Relative Cost
Austenite	Excellent	Good	Face Centered Cubic	Mid
Ferrite	Limited	Ok	Body Centered Cubic	Low
Martenite	Poor	Excellent	Body Centered Cubic	Low
Duplex	Excellent	Good	Face Centered Cubic & Body Centered Cubic	High
Super Austenite	Excellent	Good	Face Centered Cubic	Mid-High

Data

Target/Label

Austenitic,

Ferritic,

Martensitic,

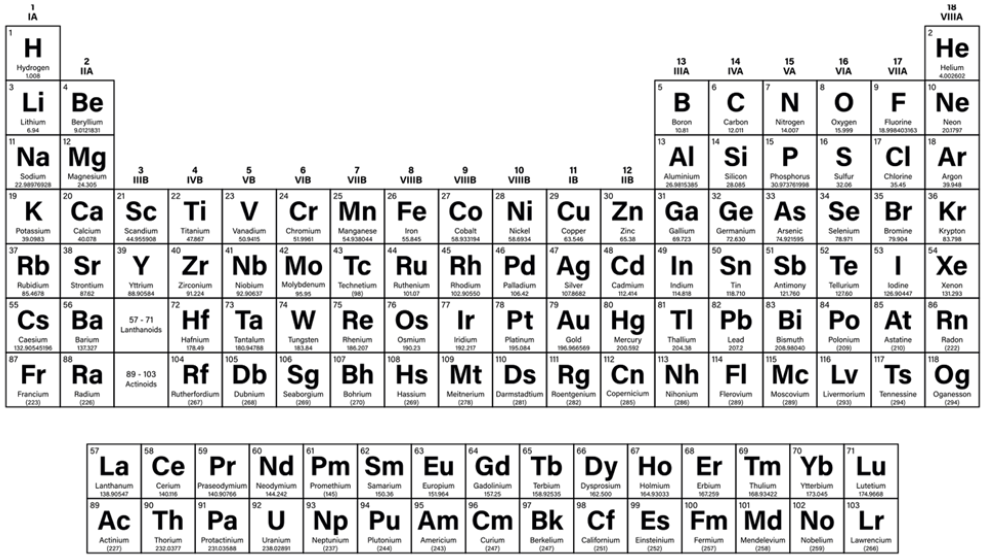
Duplex,

Super Austenitic

Features

Percentage of required

of each element

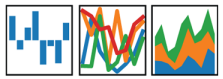


Tools

python™

pandas

$y_i t = \beta' x_{it} + \mu_i + \epsilon_{it}$



scikit learn

+ a b | e a u
S O F T W A R E

Data Extraction

Web scraping via
Beautiful Soup
Selenium

Cleaning

Numpy
Pandas

Modeling

K Nearest-Neighbors
Logistic Regression
Random Forest Gaussian
Support Vector Machines
XGBoostClassifier

Visualization

Tableau

Methodology

Metric: fbeta_score (beta= 0.5)

Scoring: Precision to minimize false positives.

Micro Averages: Aggregate the contributions of all classes to compute the average metric.

Validation: KFold(cv=5)

Step 01

Running Selected Models with Default Parameters

Step 02

Balanced the Training Data via Adasyn/ Smote

Step 03

Tune the Hyperparameters of the Model

Step 04

Compared the Validation Scores Across Best Scores of Each Model

Step 05

Chose Model Based on Score, Interpretability & Cost

Model Results

XGBoost Classifier →

Initial (Default)
Initial + SMOTE
Initial + Adasyn
GridSearch + SMOTE
GridSearch + Adasyn

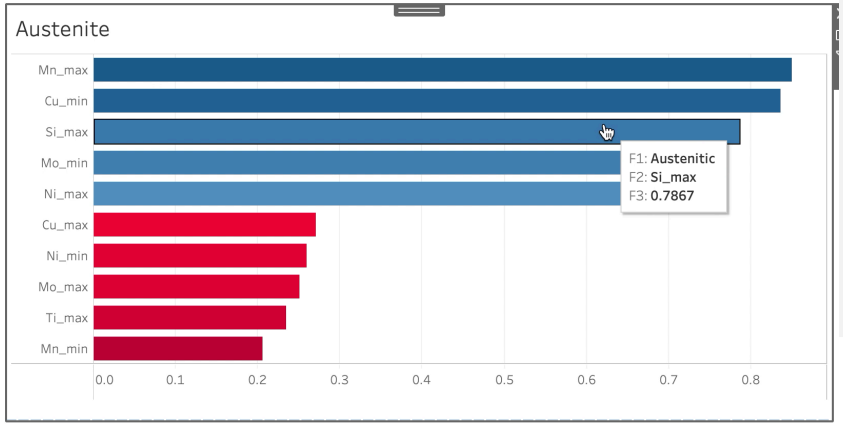
Best Validation Scores From Each Model

Model	K Nearest Neighbors	**Logistic Rogression**	Random Forest	SVM	XGBoost Classifier
Method	SMOTE + Gridsearch	SMOTE + Gridsearch	ADASYN + Gridsearch	SMOTE + Default Param	ADASYN + Gridsearch
Validation Score	0.964	0.945	0.975	0.944	0.976
Predict Score	0.825	0.835	0.864	0.870	0.874

Logistic Regression & Coefficients

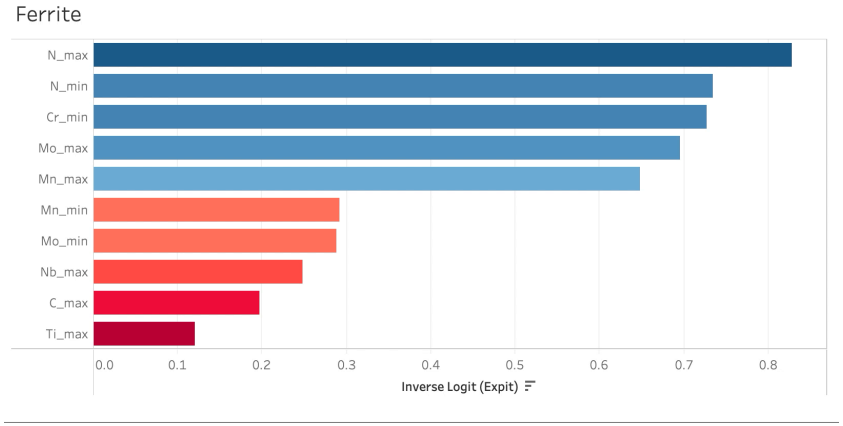
Top 5

Bottom 5



Top 5

Bottom 5



Conclusion

All models did well, but XGBoost performed the best

Balanced models perform better in classification

Future Work

Add more steel grades and add mechanical properties of steel

Perform a similar model with chemicals & polymers if dataset is available

Contact

jamesleefan@gmail.com

https://github.com/jimmfan/


Thank
You
www.website.com

References

Python

```
def kfold_test(model, X_train, y_train, n_val=5, shuffle_val=True, random_val=42,
average_val='micro', beta_val = 0.5):
    kf = KFold(n_splits=n_val, shuffle=shuffle_val, random_state=random_val)
    model_results = []
    for train_ind, val_ind in kf.split(X_train,y_train):


        X_train_k, y_train_k = X_train[train_ind], y_train[train_ind]
        X_val, y_val = X_train[val_ind], y_train[val_ind]


        model.fit(X_train_k, y_train_k)
        model_results.append(fbeta_score(model.predict(X_val), y_val, average=average_val,
beta=beta_val))
    return model_results
```

References (cont)

```
# Logistic Regression

#GridSearch
from scipy.stats import uniform
# Create regularization penalty space
penalty = ['l1', 'l2']

# Create regularization hyperparameter distribution using uniform distribution
C = uniform(loc=0, scale=50)

# Create hyperparameter options
hyperparameters = dict(C=C, penalty=penalty)
```

References (cont)

```
# SMOTED training set
grid_srch = RandomizedSearchCV(model, hyperparameters, n_iter=50, cv=5, scoring='precision_micro', iid=False)
grid_srch.fit(X_smoted, y_smoted)
print(grid_srch.best_score_)
param = grid_srch.best_params_
# {'C': 39.756662587546835, 'penalty': 'l2'}

model = LogisticRegression(
        penalty=param['penalty'],
        dual=False,
        tol=0.0001,
        C=param['C'],
        max_iter=100,
        multi_class='ovr',
        n_jobs=-1)
gridsearch_res = kfold_test(model, X_smoted, y_smoted, n_val=5, shuffle_val=True, random_val=42)
np.mean(gridsearch_res)

0.9450777202072539

model.fit(X_smoted, y_smoted)
fbeta_score(model.predict(X_test), y_test, average='micro', beta=0.5)
```