Reinvent your classifiers!

Tricks from credit risk scoring to build better classifiers

9 July 2019

Martijn Wieriks

Email: martijn@julo.co.id
Kaggle: https://www.kaggle.com/mwieriks
Github: https://github.com/sepam

Outline of talk

kaggle DAYS MEETUP

Introduction

My Kaggle and professional experience

UNESCO-IHE Institute for Water Education

accenture

SEPAM CONSULTANTS Systems Engineering. Policy. Management.

DataKind

Google AI

JULO

xendit

LOGICAI

kaggle

Motivation

Combine Kaggle with applied, real-world (business) problem-solving to become a great Data Scientist and modeler

Why Kaggle will NOT make you a great data-scientist

Want to be an Eagle or Kaggle data scientist ?

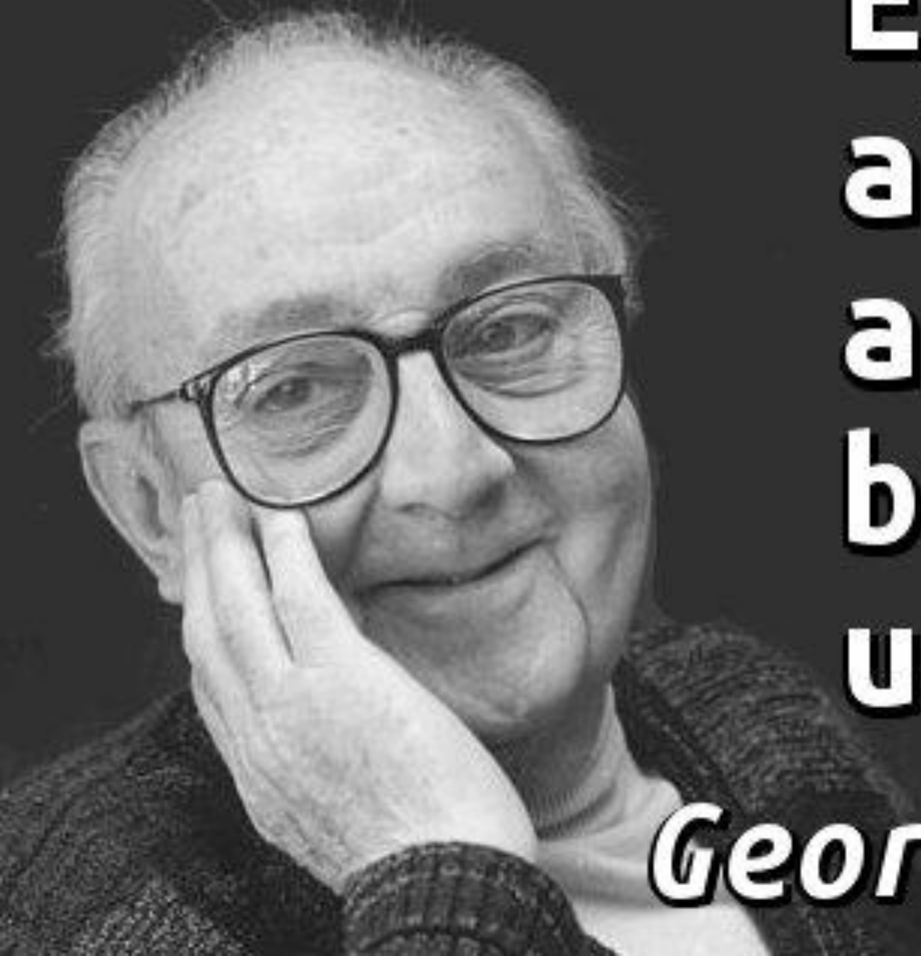
Pranay Dave Follow
Dec 7, 2018 · 6 min read ★

The Real World is not a Kaggle Competition

Michał Marcinkiewicz
Nov 30, 2018 | 17 min read Python Machine Learning r&d deep learning

No, Kaggle is unsuitable to study AI & ML. A reply to Ben Hamner

Mostapha Benhenda Follow
Apr 27, 2017 · 6 min read



"Essentially, all models are wrong, but some are useful."

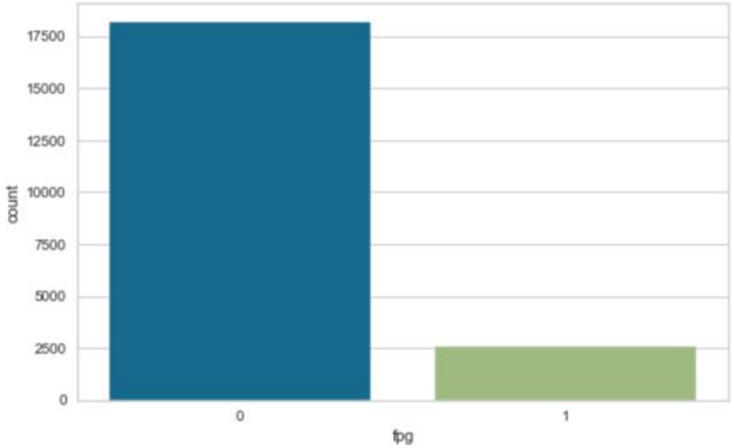
George E. P. Box

Yelo

Case study

Binary classification to predict probability of loan default at YELO - a up and coming fintech lender

```
[23]: fig, ax = plt.subplots(figsize=(8, 5))
      sns.countplot(y_train)
      plt.show()
      y_train.value_counts(normalize=True)
```




```
[23]: 0    0.874603
      1    0.125397
```

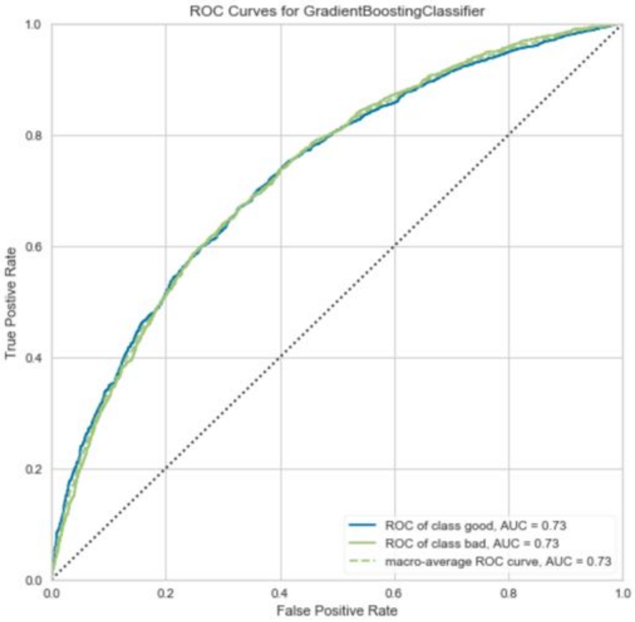
Data Preprocessing Pipeline

[18]:

```
# Data Preprocessing Pipeline

# select the columns that I want to push through the different transformers
cat_cols = list(X_train.select_dtypes(include=['object']))
num_cols = list(X_train.select_dtypes(exclude=['object']))

# categorical
cat_si = ('si', SimpleImputer(strategy='constant', fill_value='-999'))
cat_steps = [cat_si]
cat_pipe = Pipeline(cat_steps)

# numerical
num_si = ('si', SimpleImputer(strategy='constant', fill_value=-999))
num_steps = [num_si]
num_pipe = Pipeline(num_steps)

# combine the pipelines into the preprocessing pipeline
num_transformers = ('num', num_pipe, num_cols)
cat_transformers = ('cat', cat_pipe, cat_cols)
all_transformers = [num_transformers, cat_transformers]
ct = ColumnTransformer(transformers=all_transformers)
```

Model Training and Evaluation

[25]:
```
# Model building
classifier = GradientBoostingClassifier(learning_rate=0.1, max_depth=3, subsample=0.8, max_features=0.2)
model_pipe = Pipeline([('data_preprocess', ct), ('classifier', classifier)])

auc_cv = cross_val_score(model_pipe, X_train, y_train, scoring='roc_auc', cv=10)
auc_cv_mean = auc_cv.mean()
auc_cv_stdev = np.std(auc_cv)

# fit model using all training data
model = model_pipe.fit(X_train, y_train)
```



[26]:
```
# predict the test set
predictions_p1 = model.predict_proba(X_test)[:,-1]
auc_test = roc_auc_score(y_test, predictions_p1)

print("cv AUC: {}".format(round(auc_cv_mean, 3)))
print("cv Gini: {}".format(AUC_to_gini(auc_cv_mean)))
print("test AUC: {}".format(round(auc_test, 3)))
print("test Gini: {}".format(AUC_to_gini(auc_test)))
```

```
cv AUC: 0.734
cv Gini: 0.47
test AUC: 0.729
test Gini: 0.46
```

Objective:

Improve credit model performance by 15%
before end of Quarter

Problem 1

No time

Limited resources

Low complexity

Options:

1. More data
2. Better algorithms
3. Tuning / Exhaustive gridsearch
4. Ensembles / Stacking

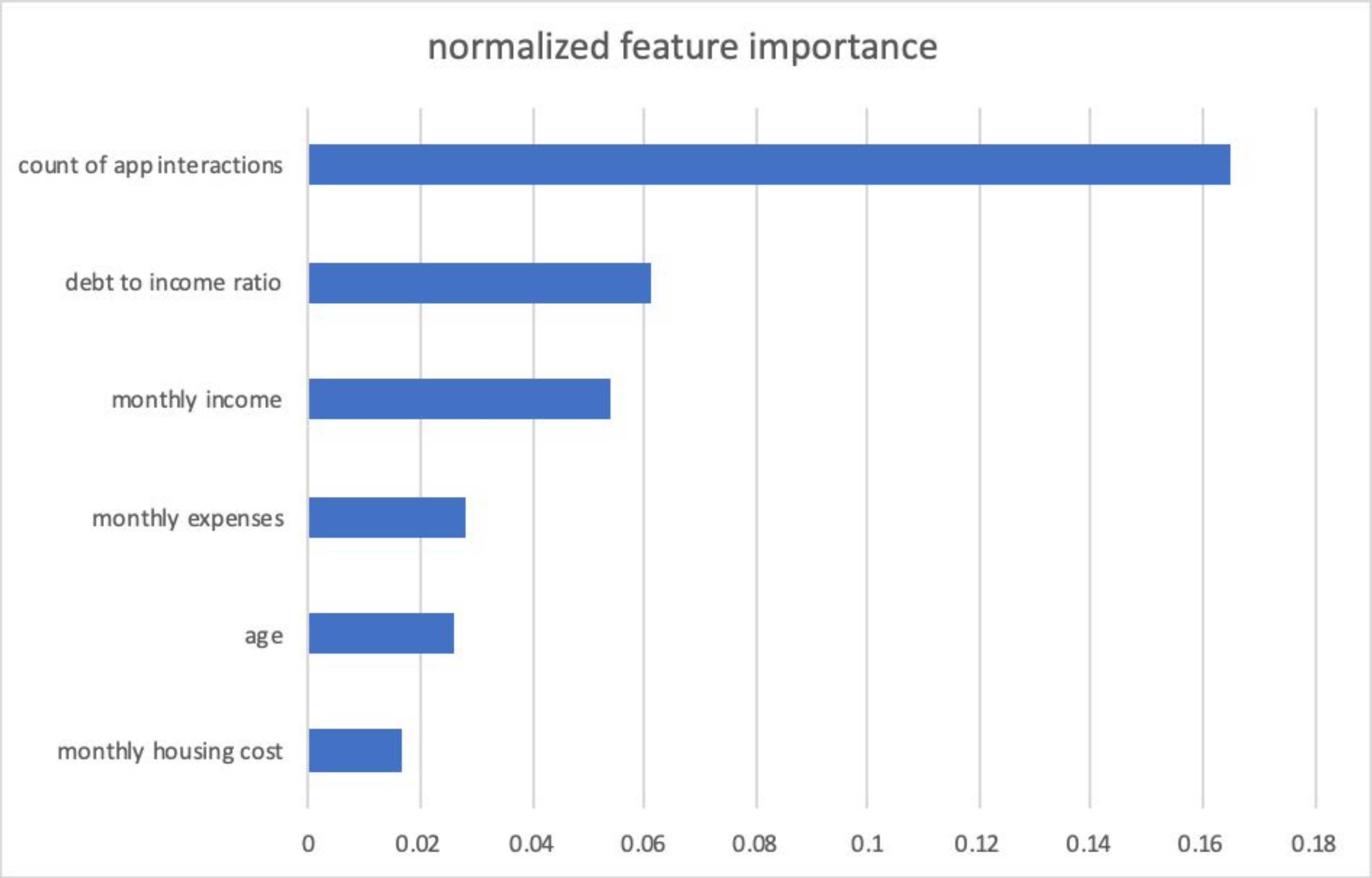
Feature engineering

category	feature idea	feature count	status	ease of building	estimated predictive value	priority	is_PR	is_done
App interaction	Count of app interactions	1	done	1	3	1	1	1
Loan purpose description text	text/NLP model prediction as feature	1	done	2	3	6	1	1
Client/Customer details	income vs years worked	1	done	3	3	9	1	1
Client/Customer details	income vs education	1	done	3	3	10	1	1
Client/Customer details	income vs age	1	done	3	3	11	1	1


Takeaway 1:
Feature engineering is *king*

Problem 2

One feature explains most variance


normalized feature importance
count of app interactions
debt to income ratio
monthly income
monthly expenses
age
monthly housing cost
0
0.02
0.04
0.06
0.08
0.1
0.12
0.14
0.16
0.18

Options:

1. Hypothesize: common sense
2. Scrutinize: data and feature code

Data leaks

Takeaway 2:

Watch out for data leaks

Problem 3

Good customers complain they get bad credit scores

Options:

Generic model behavior:

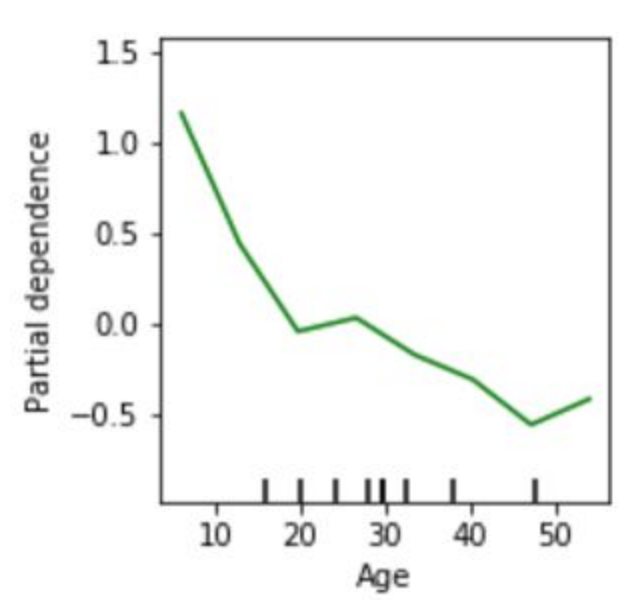
1. Model feature importance

2. Partial dependence plots

Individual prediction explanations:

1. Investigate features manually

2. Lime or SHAP



Model explanation - SHAP (SHapley Additive exPlanations)

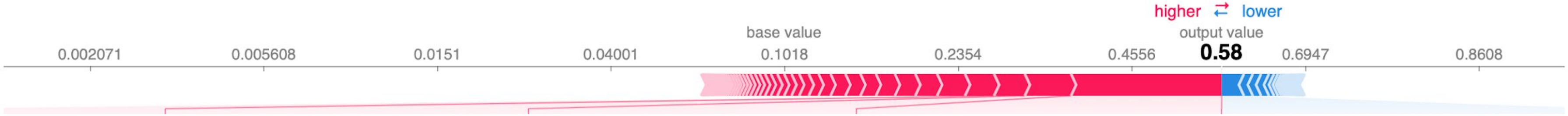
```
* title: A Unified Approach to Interpreting Model Predictions
* author: Lundberg, Scott M and Lee, Su-In
* year: 2017
* url: http://papers.nips.cc/paper/7062-a-unified-approach-to-interpreting-model-predictions.pdf
```

[*]:
```
import shap
shap.initjs()

# SHAP requires the original tree model plus the transformed data set
gbm_model = model.get_params()['classifier']
xt = model.get_params()['data_preprocess']
X_train_trans = pd.DataFrame(xt.fit_transform(X_train), columns=X_train.columns)

# explain the model's predictions using SHAP values
# same syntax works for LightGBM, CatBoost and sklearn models)
explainer = shap.TreeExplainer(gbm_model)
shap_values = explainer.shap_values(X_train_trans)
```

example: bad customer

[29]:
```
def explain_indiv(sample_index):
    return shap.force_plot(explainer.expected_value, shap_values[sample_index,:],
                           X_train_trans.iloc[sample_index, :], link='logit')

explain_indiv(181)
```

[29]:



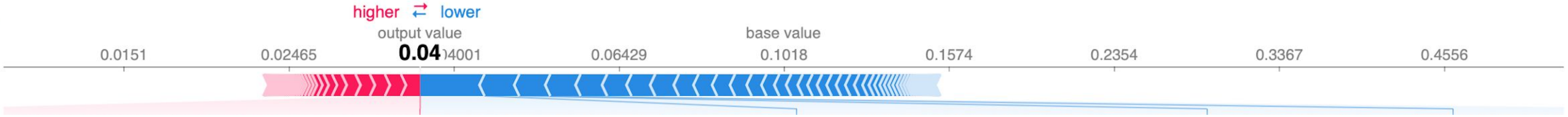
Dominating "**bad**" feature:
other_credit_application_count

Dominating "**good**" feature:
past_credit_performance

example: good customer ¶

[30]:
```
def explain_indiv(sample_index):
    return shap.force_plot(explainer.expected_value, shap_values[sample_index,:],
                           X_train_trans.iloc[sample_index, :], link='logit')

explain_indiv(2)
```

[30]:



Dominating "**bad**" feature:
debt_to_income_ratio

Dominating "**good**" feature:
past_credit_performance

Takeaway 3:

Transparency matters

Problem 4

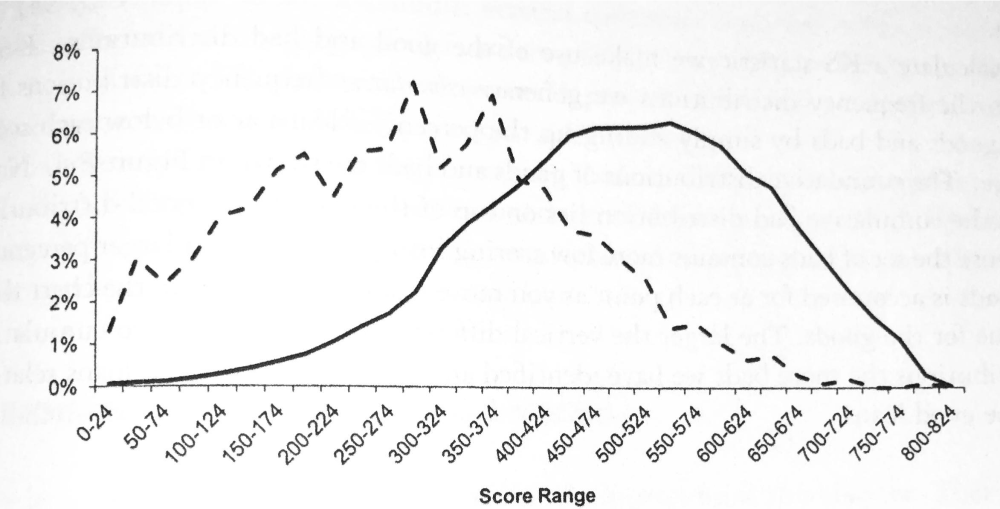
Model performance drops immediately after deployment

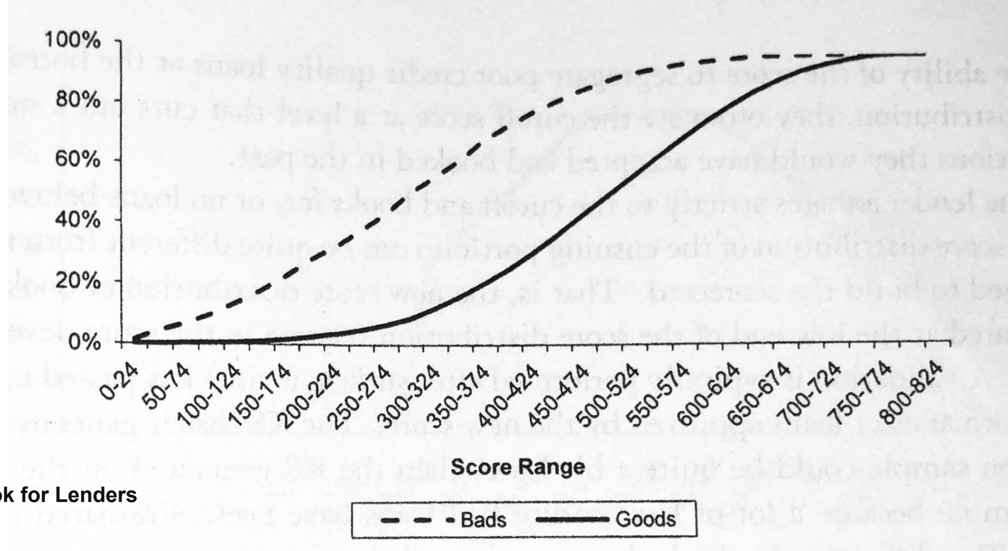
Options:

1. Roll-back & Re-model
2. Pray it gets better
3. Investigate & look at performance from more angles

Evaluation metric:

1. KS (kolmogorov smirnov) statistic
2. Measures max distance between cumulative Good vs cumulative Bad score distributions



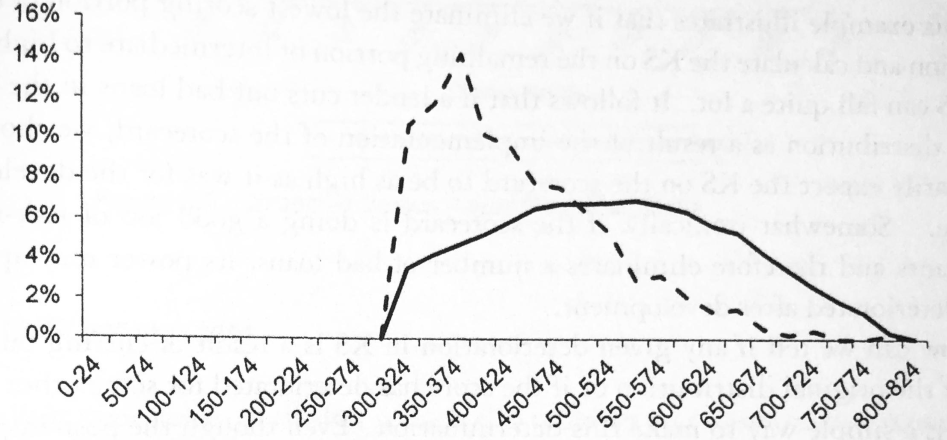


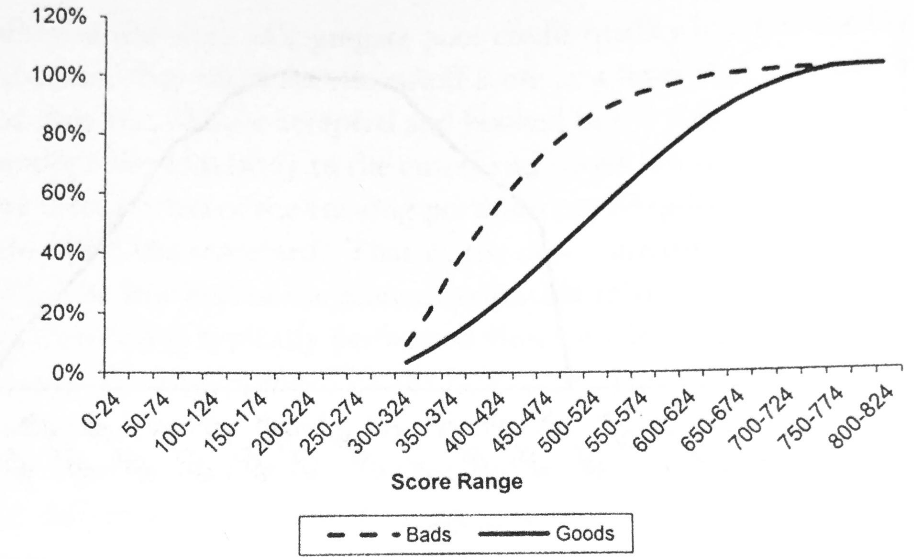
Source: Mays, E. Lynas, N. (2011) **Credit Scoring for Risk Managers: The Handbook for Lenders**

Production behavior:

1. Model cuts out applicants below score of 300

2. KS score is lower than expected!

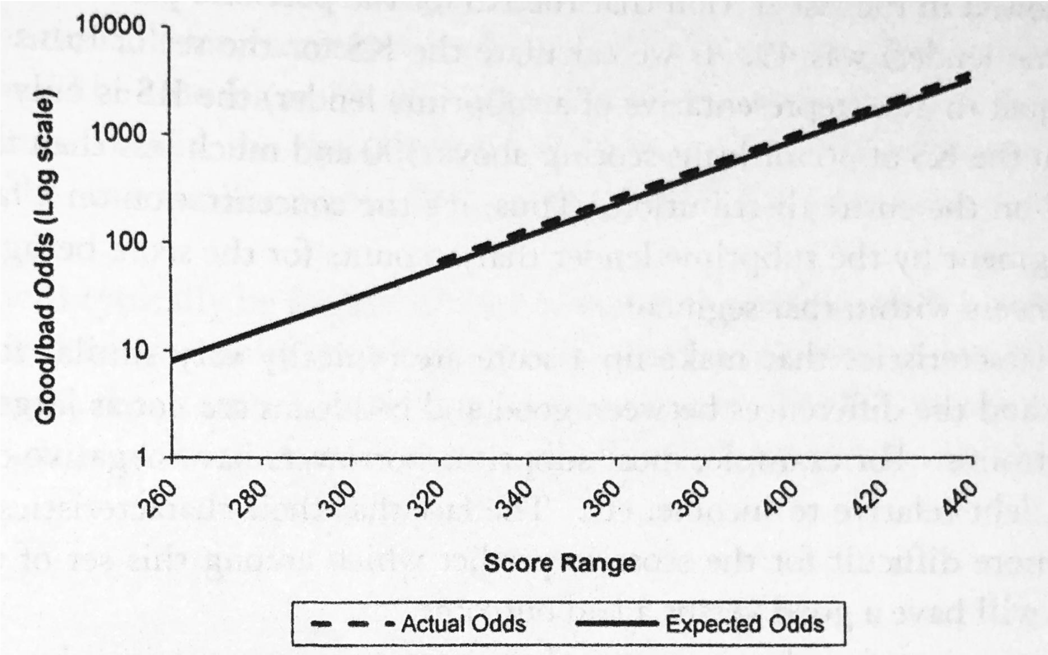




Solution:

1. Define a new metric: *change in slope measure*

2. Model actually still performs as well as it was designed, but key performance metric (KS) is lower!



Takeaway 4:

Understand your metrics!

Conclusions

1. ***Feature engineering is King***: they are the true way to differentiate and give you the edge

2. ***Always look for data leaks:*** they will harm your model or win the Kaggle competition

3. ***Transparency matters:*** understanding the weaknesses of your model and predictions allows you to tailor performance

4. ***Understand your metrics:*** Use multiple metrics to evaluate your model. Understand their strengths and weaknesses

Next steps

On learning to become better at competitive data science
- https://www.coursera.org/learn/competitive-data-science

On feature engineering:
- https://www.kaggle.com/willkoehrsen/introduction-to-manual-feature-engineering

- https://www.kdnuggets.com/2018/11/secret-sauce-top-kaggle-competition.html

- https://medium.com/comet-ml/manual-feature-engineering-kaggle-home-credit-db1362d683c4

On data leakages:
- https://www.kaggle.com/dansbecker/data-leakage

On explainability:
- Dan Becker's Free Micro Course on ML Explainability:
https://www.kaggle.com/learn/machine-learning-explainability

On model metrics:
- https://www.analyticsvidhya.com/blog/2016/02/7-important-model-evaluation-error-metrics/