Preparing Messy Real World Data For Supervised Machine Learning

Basic through Advanced Techniques

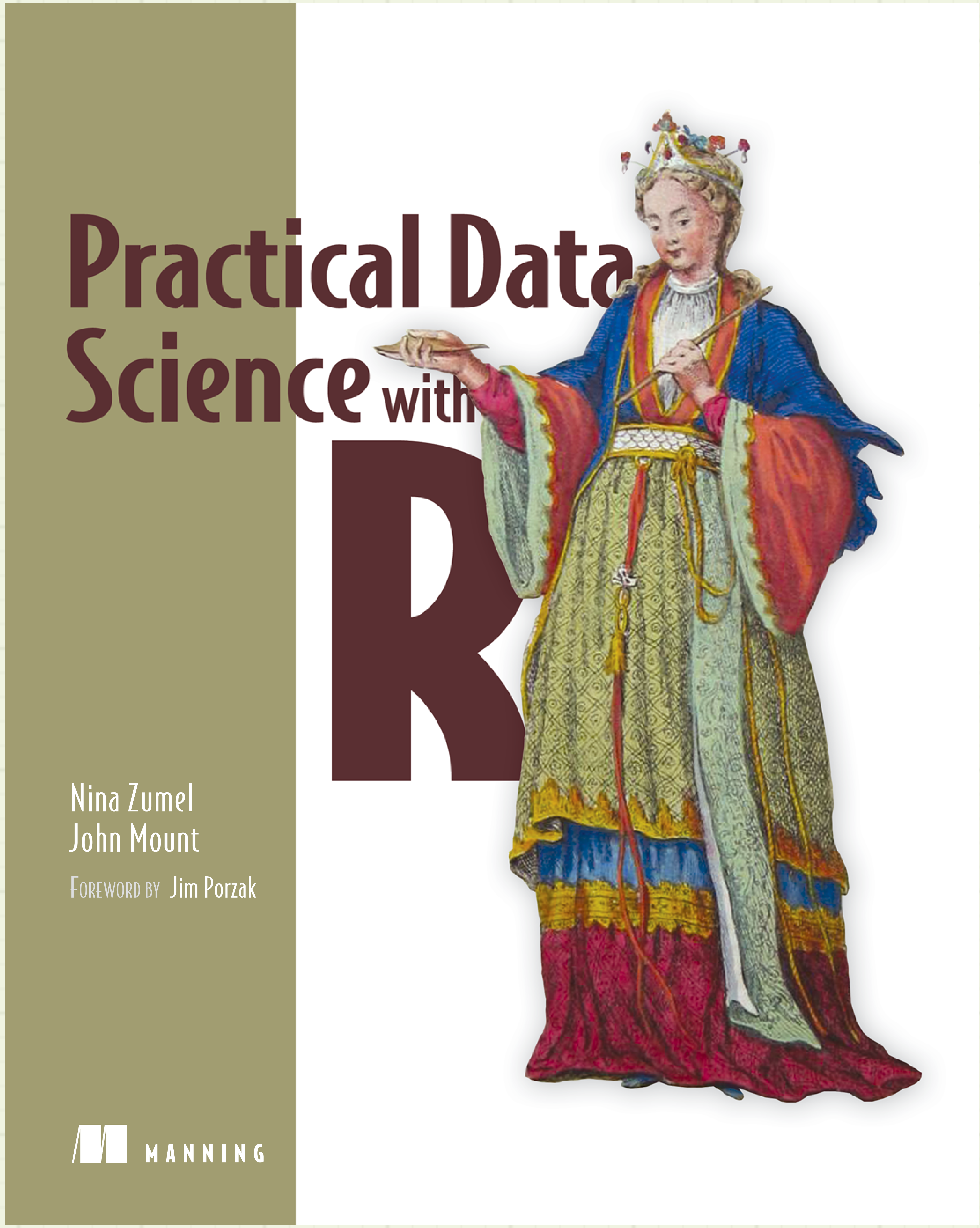
(or: Introduction to the `vtreat`)

Nina Zumel
John Mount
Win-Vector, LLC
http://www.win-vector.com/

Win-Vector LLC

Who we are

- Nina Zumel and John Mount
- Principal Consultants at Win-Vector LLC
- Authors of Practical Data Science with R



Outline

- Data Preparation
  - Typical data problems & possible solutions
- `vtreat`: Automating variable treatment in R and Python
- Examples of automated variable treatment
- Conclusion

Only URL to remember, these slides:
https://github.com/WinVector/Examples/blob/master/pyvtreat/vtreat.pdf

Throughout this talk

- We will keep an idealized goal in mind: using machine learning to build a predictive model.
- We assume we can delegate the modeling or machine learning to a library, and take on the responsibility for data preparation and cleaning.
- Having a single ideal goal allows us to apply seemingly “ad-hoc” fixes in a principled manner.
    - We can check if our “fixes” are for good or bad.
    - We are not limited to mindlessly combining prior “name brand” procedures.

Data Preparation

Why Prepare Data at All?

- To facilitate modeling/analysis
  - Clean dirty data
  - Format data the way machine learning algorithms expect it
- Not a substitute for getting your hands dirty
  - But some issues show up again and again

Win-Vector LLC

Typical Data Problems

- “Bad” numerical values (NA, NaN, sentinel values)
- Categorical variables: missing values, missing levels
- Categorical variables: too many levels
- Invalid values
    - Out of range numerical values
    - New/invalid category levels

First Issue: Bad/missing Numeric Values

Bad Numerical Values

Miles driven	Gas Consumption
100	2
235	0
150	7.5
200	5.5
0	0
300	NA

MPG	
50	
Inf	Electric car/bad calculation
20	
36.4	
NaN	Non-numeric typo/ bad calculation
NA	Electric car

Win-Vector LLC

Whither Bad Values?

- “Faulty Sensor” — values are missing at random
    - Assume they come from the same distribution as the other values
    - The mean of the “good” values is a reasonable stand-in
- Systematically missing
    - Electric cars
    - They WILL behave differently from gas or hybrid cars
    - The mean of the good values is not a valid stand-in

Win-Vector LLC

A number of possible solutions

- Naive: skip rows with missing values
- Multiple models: build many models using incomplete subsets of the columns.
- Imputation: build additional models that guess values for missing variables based on other variables.
- Statistical: sum-out or integrate-out missing values.
- Pragmatic: replace with harmless stand-ins and add notation so the machine learning system is aware of the situation.

Missingness as signal

- In business analytics missing data is often an indicator of where the data came from and how it was processed.
- Consequently it is often one of your more informative signals when modeling!

One Pragmatic Solution

MPG
50
Inf
20
36.4
NaN
NA

MPG	MPG_isBad
50	FALSE
35.5	**TRUE**
20	FALSE
36.4	FALSE
35.5	**TRUE**
35.5	**TRUE**

Second Issue: Unexpected or Novel Categorical Levels

Categorical Variables:
Missing Values and Novel Levels

TrainingData

Residence
CA
NV
OR
CA
CA
NA
WA
OR
WA

NewData

Residence
NV
OR
NV
WY
CA
CA
NV
NA
OR

Win-Vector LLC

Novel Levels - Model Failure

```
#R
model = lm("premium~age+sex+residence",
            data=TrainingData)

predPremium = predict(model,
                      newdata=NewData)

Error in model.frame.default(Terms, newdata,
na.action = na.action, xlev = object$xlevels) :
factor residence has new levels WY
```

Win-Vector LLC

On the Way to the Solution: Indicator Variables

Residence
CA
NV
OR
CA
CA
NA
WA
OR
WA

Res_NA	Res_CA	Res_NV	Res_WA	Res_OR
0	1	0	0	0
0	0	1	0	0
0	0	0	0	1
0	1	0	0	0
0	1	0	0	0
1	0	0	0	0
0	0	0	1	0
0	0	0	0	1
0	0	0	1	0

- Common implementations.
    - R:
        - model.matrix()
    - Python:
        - pandas.get_dummies()
        - sklearn.preprocessing.OneHotEncoder()
- We recommend:
    - vtreat

Win-Vector LLC

Three Possible Solutions

Notional Training Data Proportions

NA	CA	NV	WA	OR
1/9	1/3	1/9	2/9	2/9

1) A novel level is weighted proportional to known levels

Residence
WY

Res_NA	Res_CA	Res_NV	Res_WA	Res_OR
1/9	1/3	1/9	2/9	2/9

2) A novel level is treated as "no level"

Residence
WY

Res_NA	Res_CA	Res_NV	Res_WA	Res_OR
0	0	0	0	0

3) A novel level is treated as uncertainty among rare levels

Residence
WY

Res_NA	Res_CA	Res_NV	Res_WA	Res_OR
1/2	0	1/2	0	0

vtreat solution

Residence	# of occurrences
CA	2000
NV	1100
OR	1000
WA	1500
ID	14
CO	8

Residence	prevalence
CA	0.36
NV	0.20
OR	0.18
WA	0.27
ID	0.002
CO	80.002

"no level" plus "prevalence code"

Residence
WY

Res_NA	Res_CA	Res_NV	Res_WA	Res_OR	Res_prevalence
0	0	0	0	0	0.0

Third Issue: Categorical Variables with Very Many Levels

Categorical variables: Too many levels

ZIP	SalePriceK
94127	725
94564	402
90011	386
94704	790
95555	1195
94109	903
94124	625
94123	439
94562	290

- Too many levels is a computational problem for some machine learning algorithms.

- You will inevitably have a novel level

The Best (but not always possible) Solution

Use as join key into domain knowledge.

San Francisco County ZIP codes	Avg. listing price Week ending Aug 13	Median sales price Date range: May-Aug '14
Name	Amount	Amount
94124	$571,667	$625,000
94134	$619,495	$640,000
94132	$713,583	$835,000
94102	$768,558	$605,000
94112	$771,234	$728,250
94111	$877,000	$959,000
94116	$904,071	$1,025,000
94107	$1,019,113	$908,500
94117	$1,057,000	$1,125,000
94131	$1,057,160	$1,200,000
94110	$1,128,511	$1,082,000
94122	$1,227,482	$930,000
94114	$1,405,793	$1,452,000
94103	$1,406,597	$850,000
94109	$1,408,431	$903,500
94105	$1,549,047	$1,107,500
94127	$1,569,846	$1,300,000

Pragmatic Solution: "Impact Coding"

ZIP	avgPriceK	ZIP_impact
90011	386	-253.4
94109	903	263.6
94124	532	-107.4
94127	960	320.6
94564	346	-293.4
94704	790	150.6
globalAvg	639.4	0

Impact-coding the ZIP variable

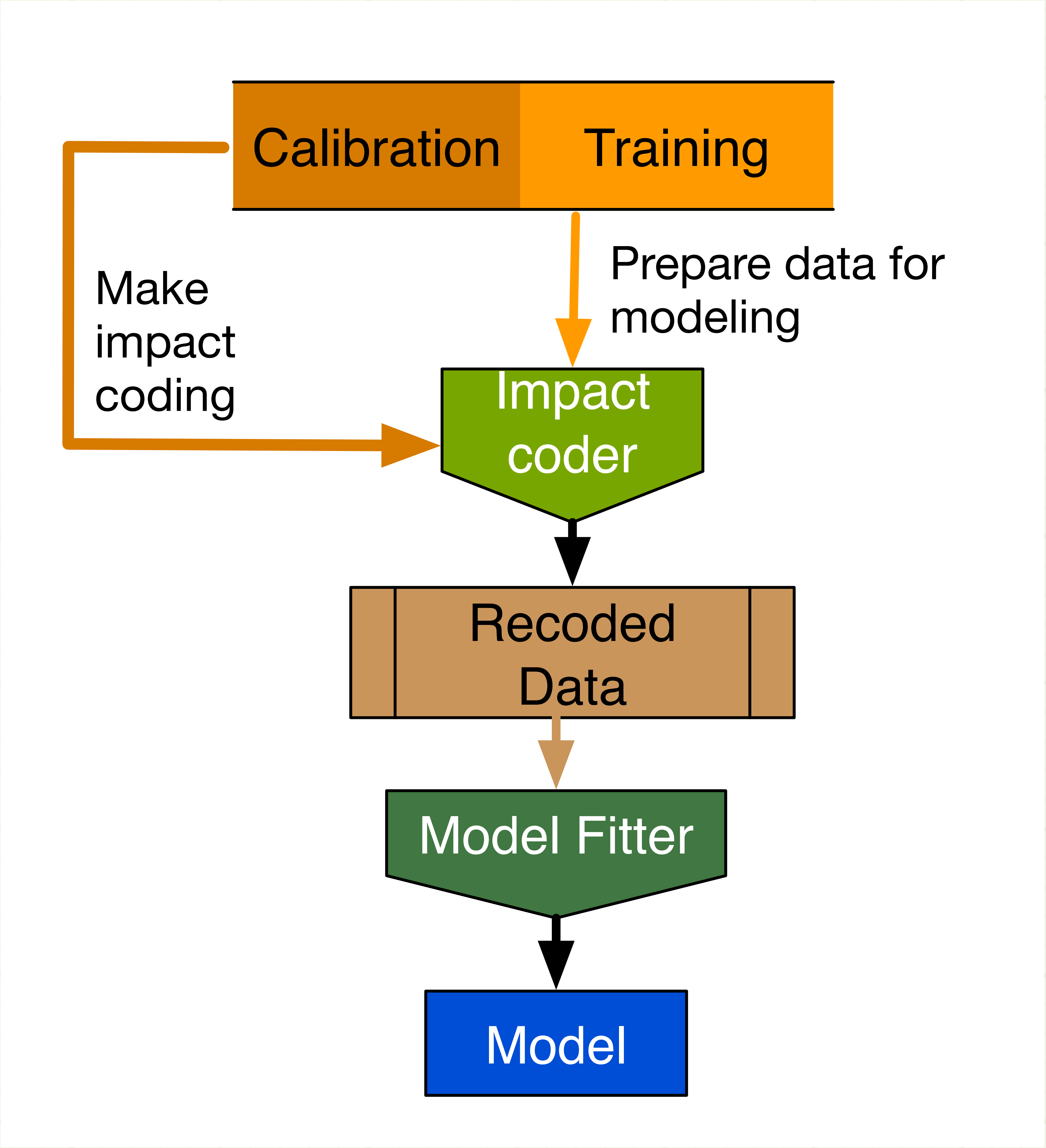
ZIP
94127
94564
90011
94704
94127
94109
94124
94124
93401

ZIP_impact
320.6
-293.4
-253.4
150.6
320.6
263.6
-107.4
-107.4
0

Win-Vector LLC

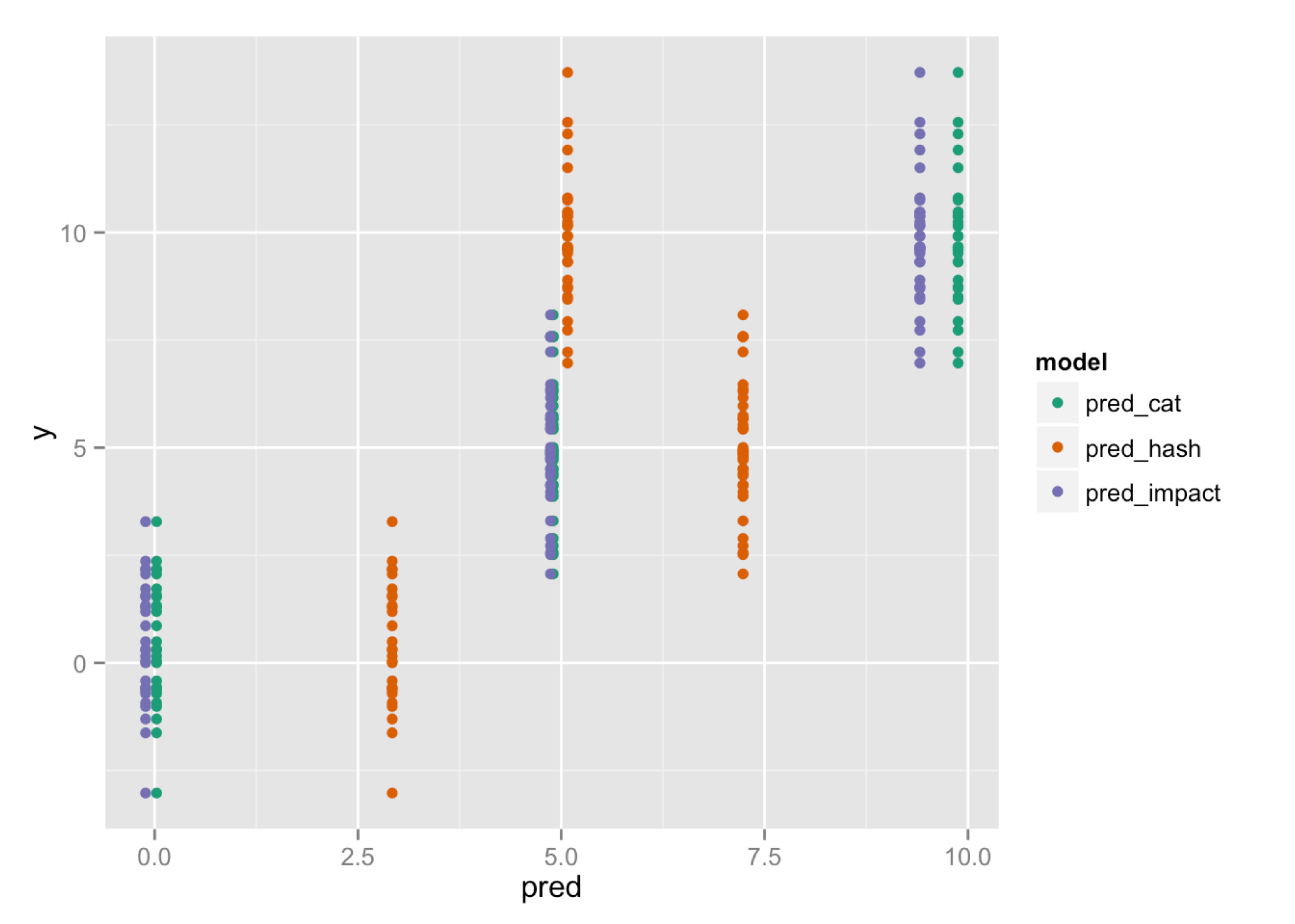
Don't Use Training Data to Impact Code!

- Can introduce undesriable nested model bias
- Full model may overestimate value of impact coded variable
- Use separate calibration data or cross-validation methods
  - vtreat has built-in cross-validation methods



Sidebar:
Impact-Code; **DON'T Hash!**

- `Python/scikit-learn`: only takes numerical variables
- Hashing loses information!



hashing idea: https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.FeatureHasher.html
why not: http://www.win-vector.com/blog/2014/12/a-comment-on-preparing-data-for-classifiers/

Example Problem

- Dataset KDD2009
    - Data set for KDD DataCup 2009
        - Task: predict account cancelation (or “churn”) from supplied features.
    - Very messy data
        - 230 columns/variables
        - 50000 instances
        - Numeric and Categorical values
        - Nonsense column and level names, and no data dictionary
        - Many missing values
        - Unbalanced task: churn rate around 7%

Win-Vector LLC

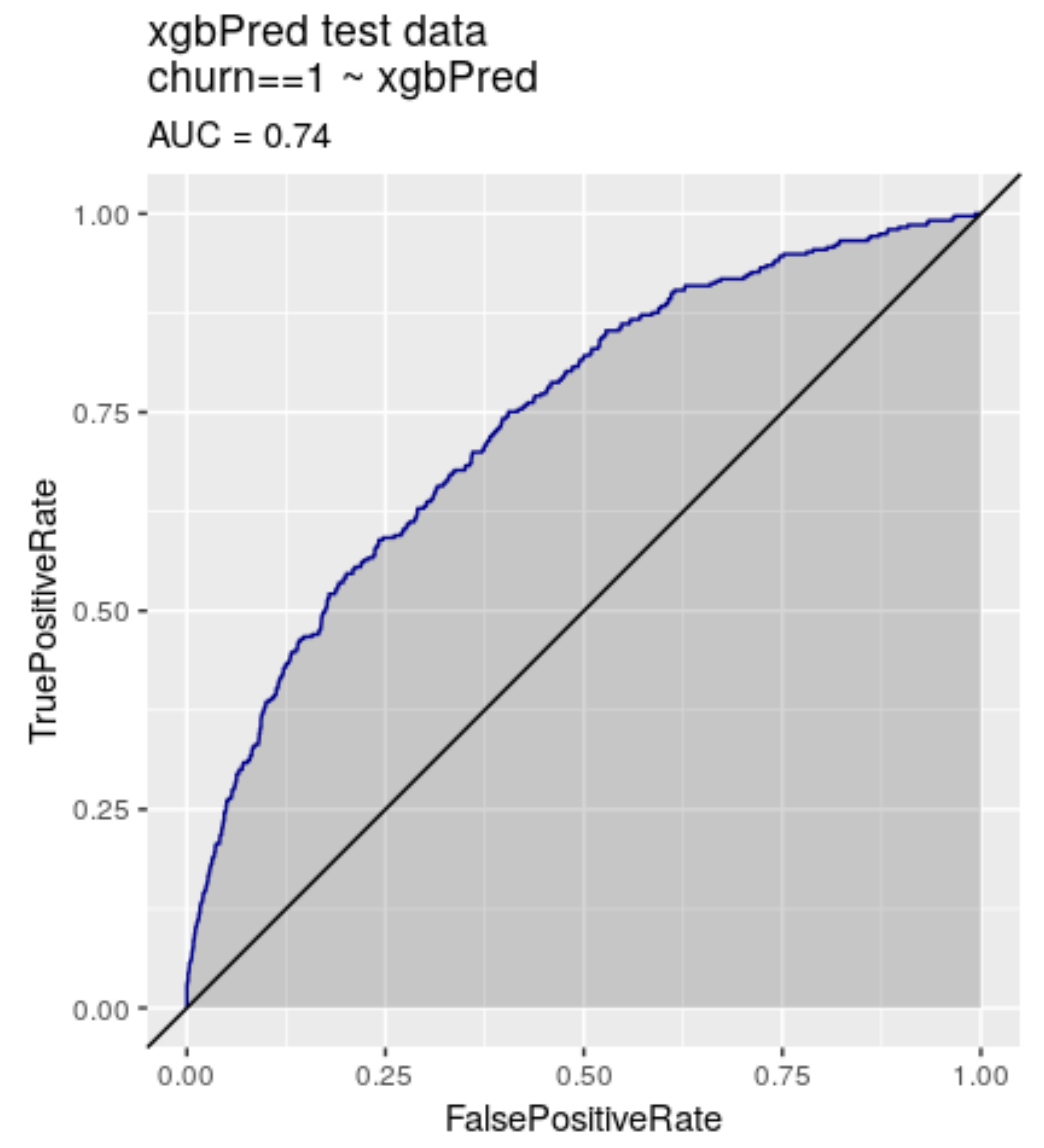
R Solution

```
library(vtreat)

# Learn data encoding
cfe = mkCrossFrameCExperiment(dTrain,
                              vars,yName,yTarget,
                              customCoders = customCoders,
                              smFactor=2.0,
                              parallelCluster=cl)

# pick variables
selvars <- cfe$treatments$scoreFrame$varName

# fit model
params <- list(max_depth = 5,
               objective = "binary:logistic",
               nthread = ncore)
model <- xgb.cv(data = as.matrix(treatedTrainM[, selvars, drop = FALSE]),
                label = treatedTrainM[[yName]],
                nrounds = 400,
                params = params,
                nfold = 5,
                early_stopping_rounds = 10,
                eval_metric = "logloss")
nrounds <- model$best_iteration
model <- xgboost(data = as.matrix(treatedTrainM[, selvars, drop = FALSE]),
                 label = treatedTrainM[[yName]],
                 nrounds = nrounds,
                 params = params)
```



https://github.com/WinVector/PDSwR2/blob/master/KDD2009/KDD2009vtreat.md

Python xgboost expects numeric types!

```
#Python
try:
    fitter.fit(d_train, churn_train)
except Exception as ex:
    print(ex)

DataFrame.dtypes for data must be int, float or
bool.
                Did not expect the data types in
fields Var191, …
```

This surprises R users who are used to implicit model.matrix() effects, but common for Python and sklearn.

Win-Vector LLC

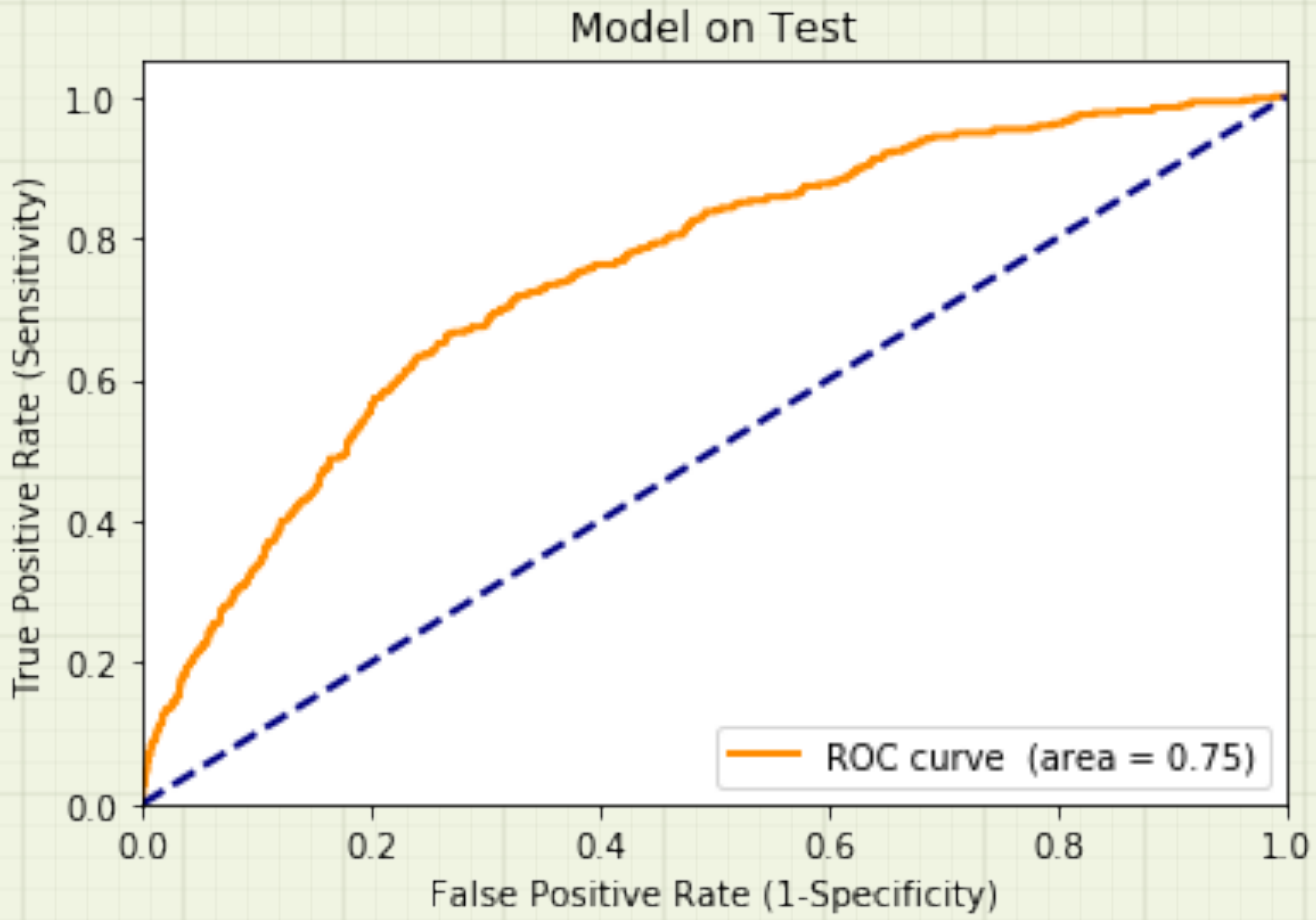
Python/Pandas Solution

```
import vtreat

# Learn data encoding
plan = vtreat.BinomialOutcomeTreatment(outcome_target=True)
cross_frame = plan.fit_transform(d_train, churn_train)

# pick variables
model_vars = numpy.asarray(plan.score_frame_[“variable"][
                plan.score_frame_["recommended"]])

# fit model
fd = xgboost.DMatrix(data=cross_frame.loc[:, model_vars],
                     label=churn_train)
x_parameters = {"max_depth":3, "objective":'binary:logistic'}
cv = xgboost.cv(x_parameters, fd,
                num_boost_round=100, verbose_eval=False)
best = cv.loc[cv[“test-error-mean"] <=
                  min(cv["test-error-mean"] + 1.0e-9), :]
ntree = best.index.values[0]
fitter = xgboost.XGBClassifier(n_estimators=ntree,
                               max_depth=3,
                               objective=‘binary:logistic')
model = fitter.fit(cross_frame.loc[:, model_vars], churn_train)
```

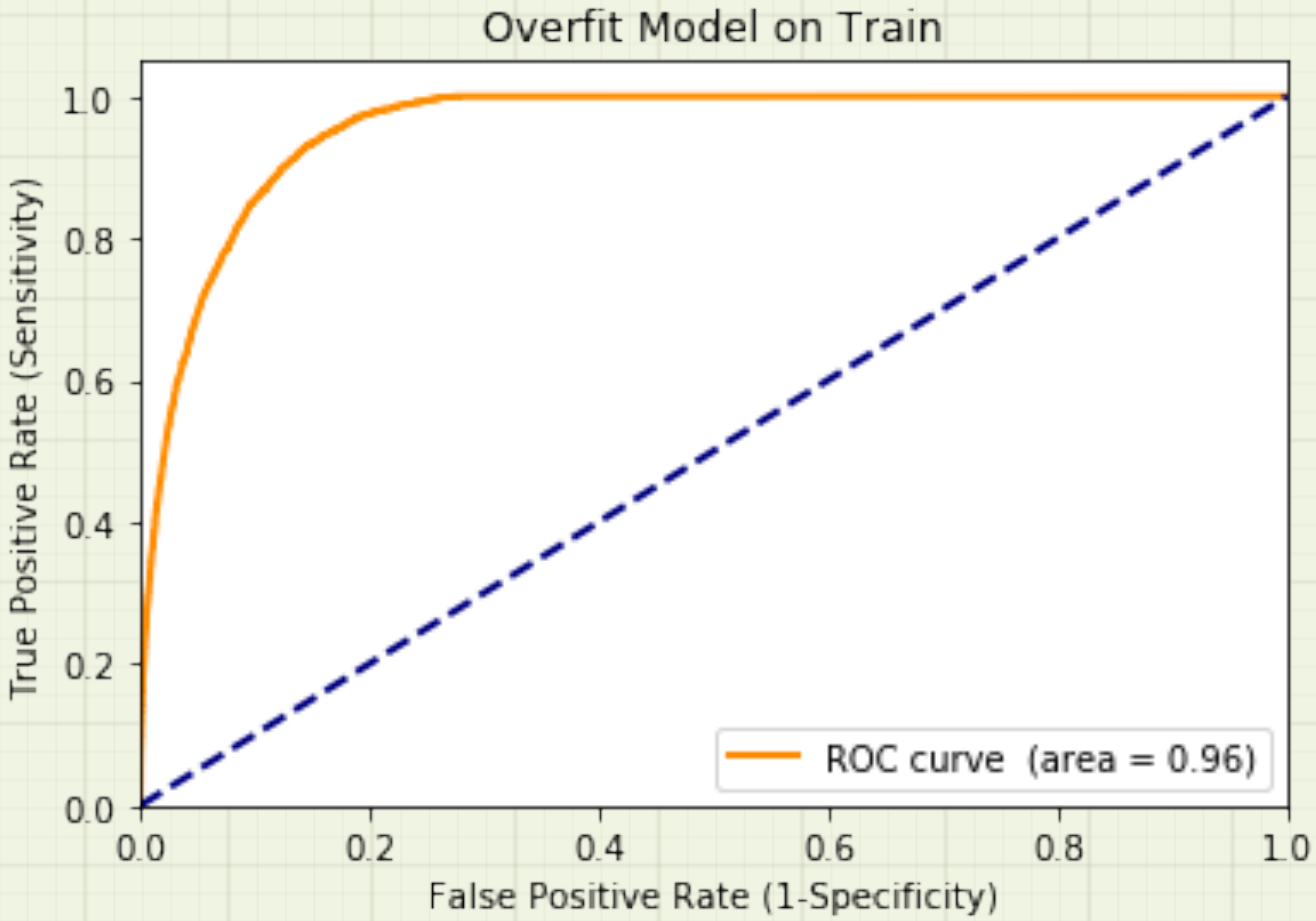


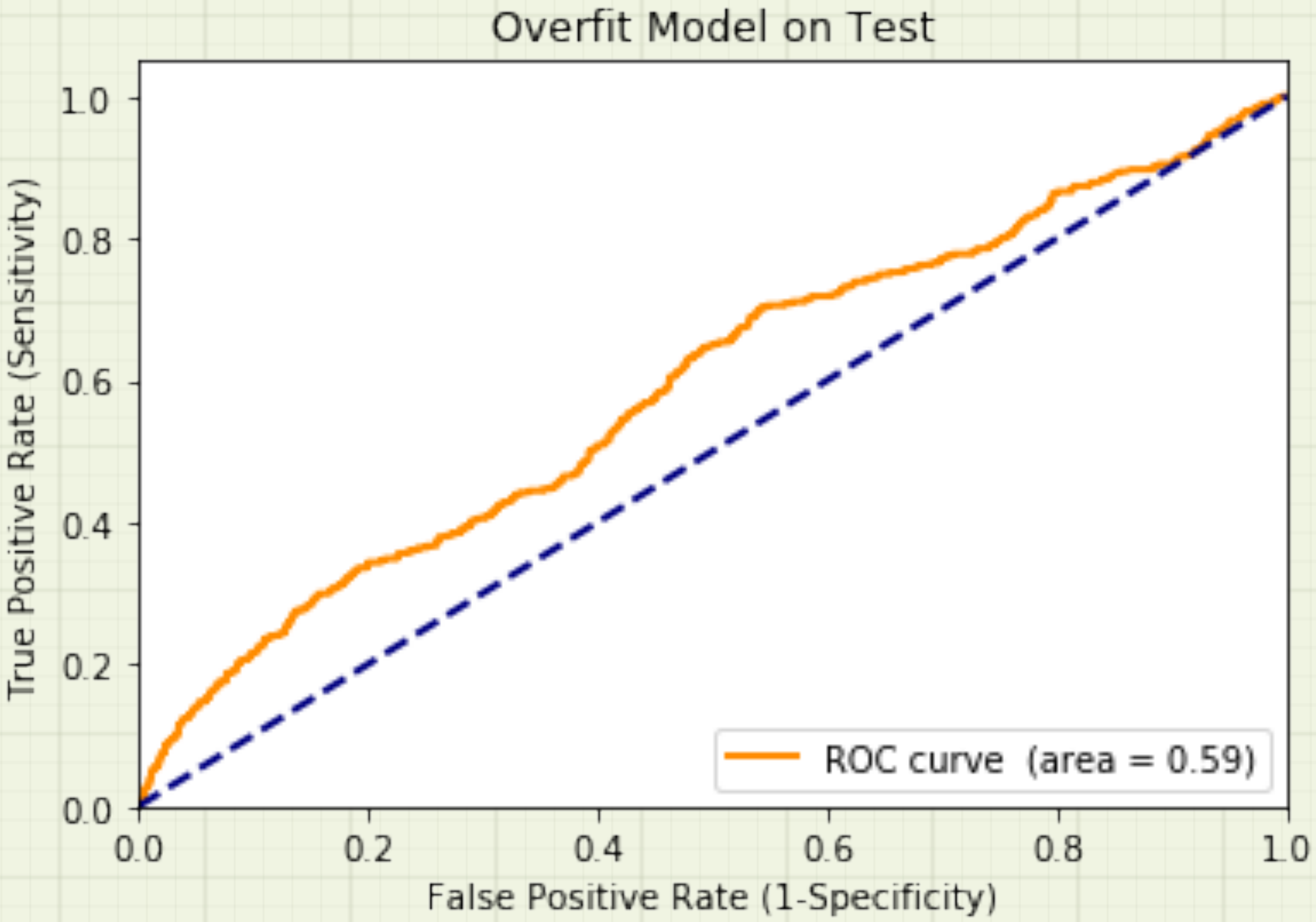
https://github.com/WinVector/pyvtreat/blob/master/Examples/KDD2009Example/KDD2009Example.ipynb

Result

- Off the shelf variable treatment, with off the shelf machine learning, and no hyper-parameter tuning, got us to within the winning AUC performance of 0.7467 (day one) to 0.7651 (end of contest) in minutes (ref: http://proceedings.mlr.press/v7/guyon09/guyon09.pdf).

Without the cross-frame methodology





Win-Vector LLC

Conclusions

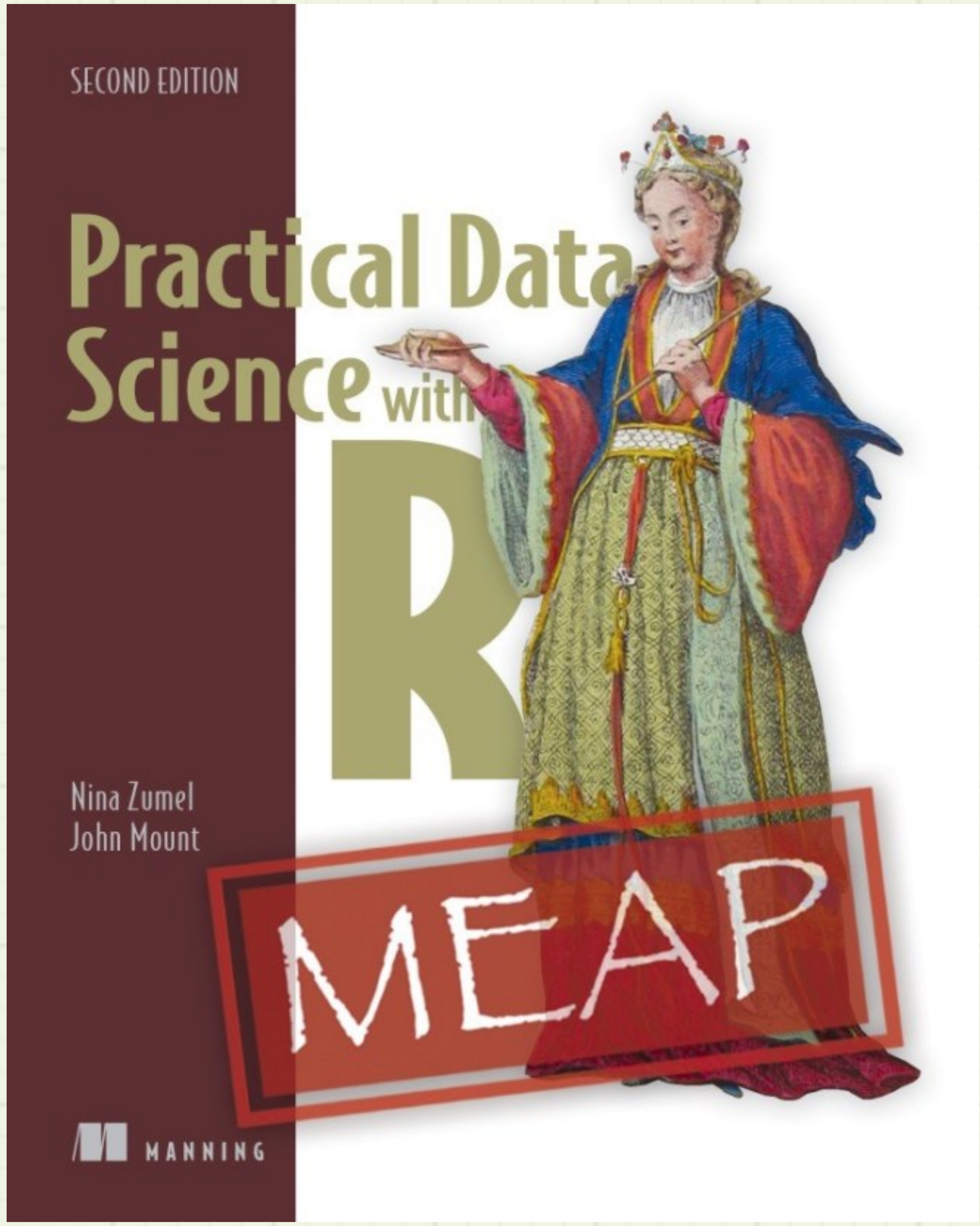
- There's no substitute for getting your hands in the data
- Nonetheless, some variable treatments are reusable again and again
- Automate what can be automated to leave the data scientist more time to develop high-value domain specific ideas

References

- Impact Coding
  - http://www.win-vector.com/blog/2012/07/modeling-trick-impact-coding-of-categorical-variables-with-many-levels/
  - http://www.win-vector.com/blog/2012/08/a-bit-more-on-impact-coding/
  - “vtreat paper” https://arxiv.org/abs/1611.09477
- R vtreat:
  - https://github.com/WinVector/vtreat/
- Python/Pandas vtreat:
  - https://github.com/WinVector/pyvtreat

Book Discount!

- http://www.win-vector.com/blog/2019/07/r-books-discount/
- 40% off!



Win-Vector LLC