Good features beat algorithms

Europython 2018 - Edinburgh 🏴
27 July 2018

Pietro Mascolo
Senior Data Scientist - Optum

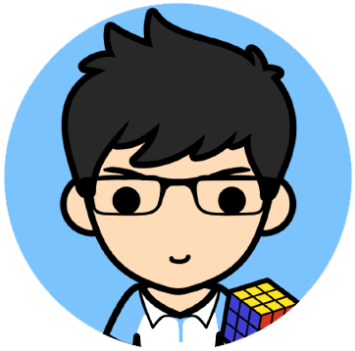
Slides and code: bit.ly/2J6xIzj (https://bit.ly/2J6xIzj)

I'm going to:

- introduce myself;
- tell a story;
- show you some code;
- show you some numbers.

2

Me



- (🇧🇷 * 🇮🇹) + 🇮🇪 + 🇮🇳;
- Physicist ⟶ Data Scientist ;
- **Python** / **Go** / **Kotlin** / Scala / ... ;
- Ham Radio Operator (**IZ4VVE**);
- Mountain hiker and Karateka;
- Amateur Photographer;
- ...

Optum / UHG

optum.com (http://www.optum.com), unitedhealthgroup.com (http://www.unitedhealthgroup.com)

4

Once upon a time, there was a brave knight...

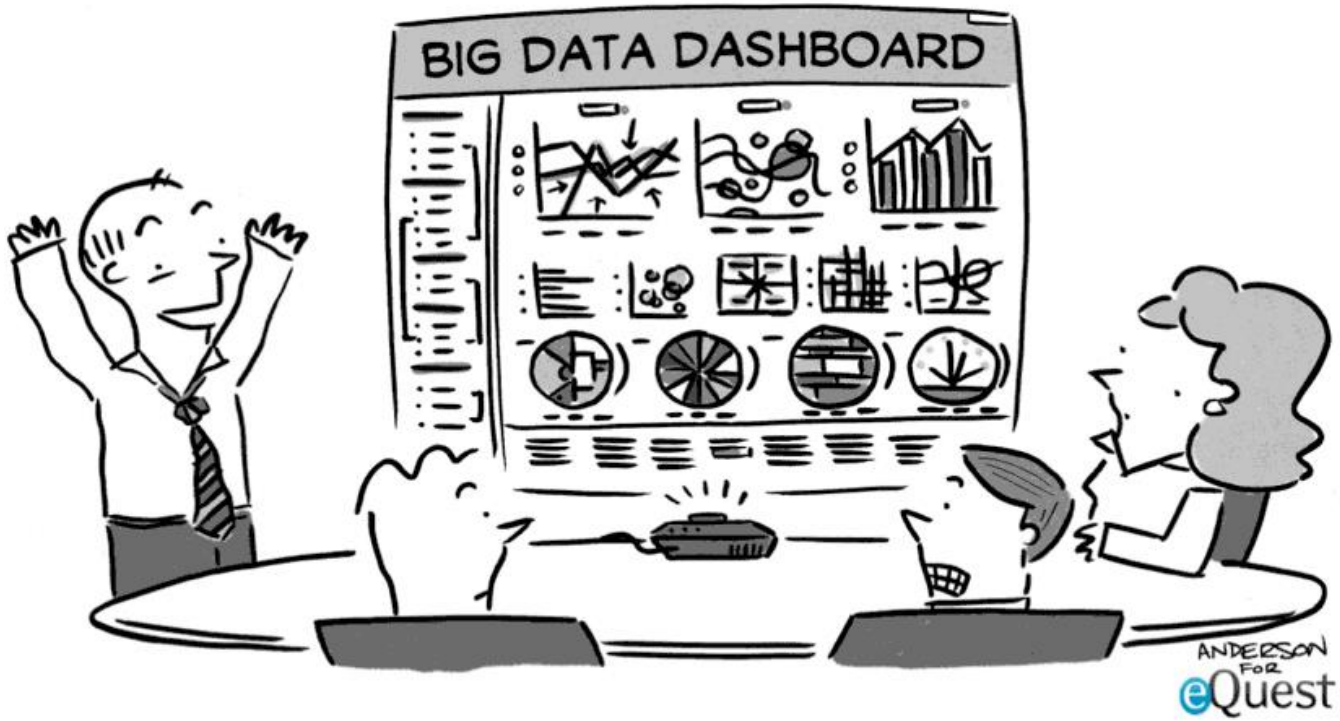
5

6

The evil overlords

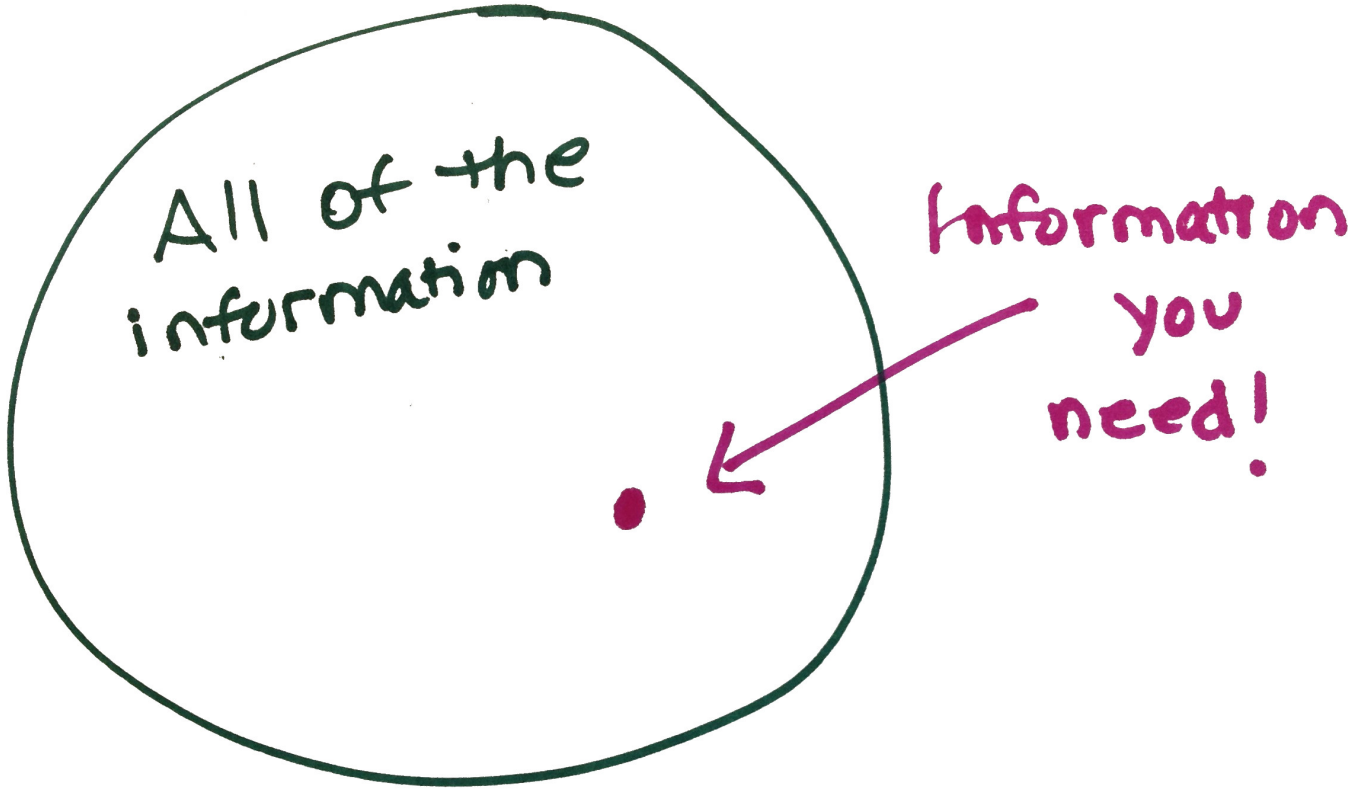
The dragon

It's just too much...



"After careful consideration of all 437 charts, graphs, and metrics,
I've decided to throw up my hands, hit the liquor store,
and get snockered. Who's with me?!"

Reality



Problems!

- Understanding the data.
- Training time.
- Hardware requirements (sometimes).
- Overfitting.
- Decreased performance.
- Leaks.
- ...

11

Sometimes you won't even notice it...

12

When you get 99.9% accuracy 😃

... 2.3 seconds later 🤔

Let's move on

15

Let's go simpler: this is just a talk after all...

We have a dataset and a target.

```
df = pd.DataFrame(data['data'], columns=data['feature_names'])
print(f"Dataset contains {df.shape[0]} rows x {df.shape[1]} columns")
```

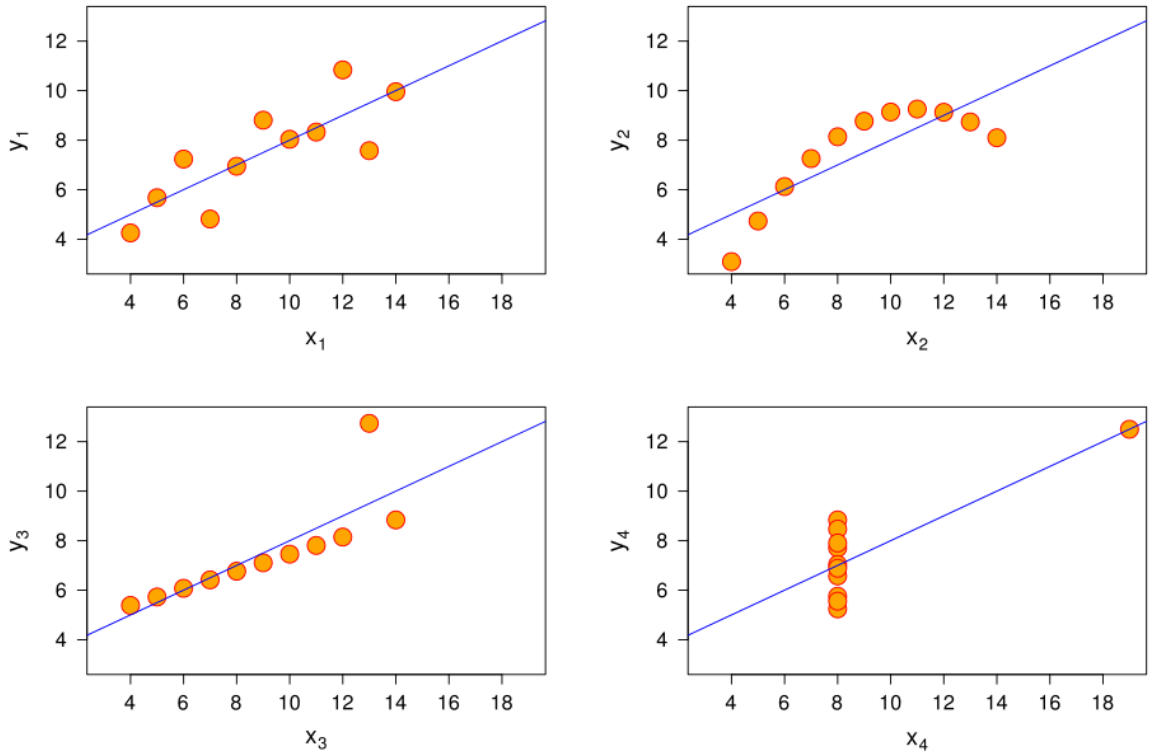
Run

16

Which features do we choose? And how do we choose them?

17

Be mindful of your metrics!



Anscombe quartet

18

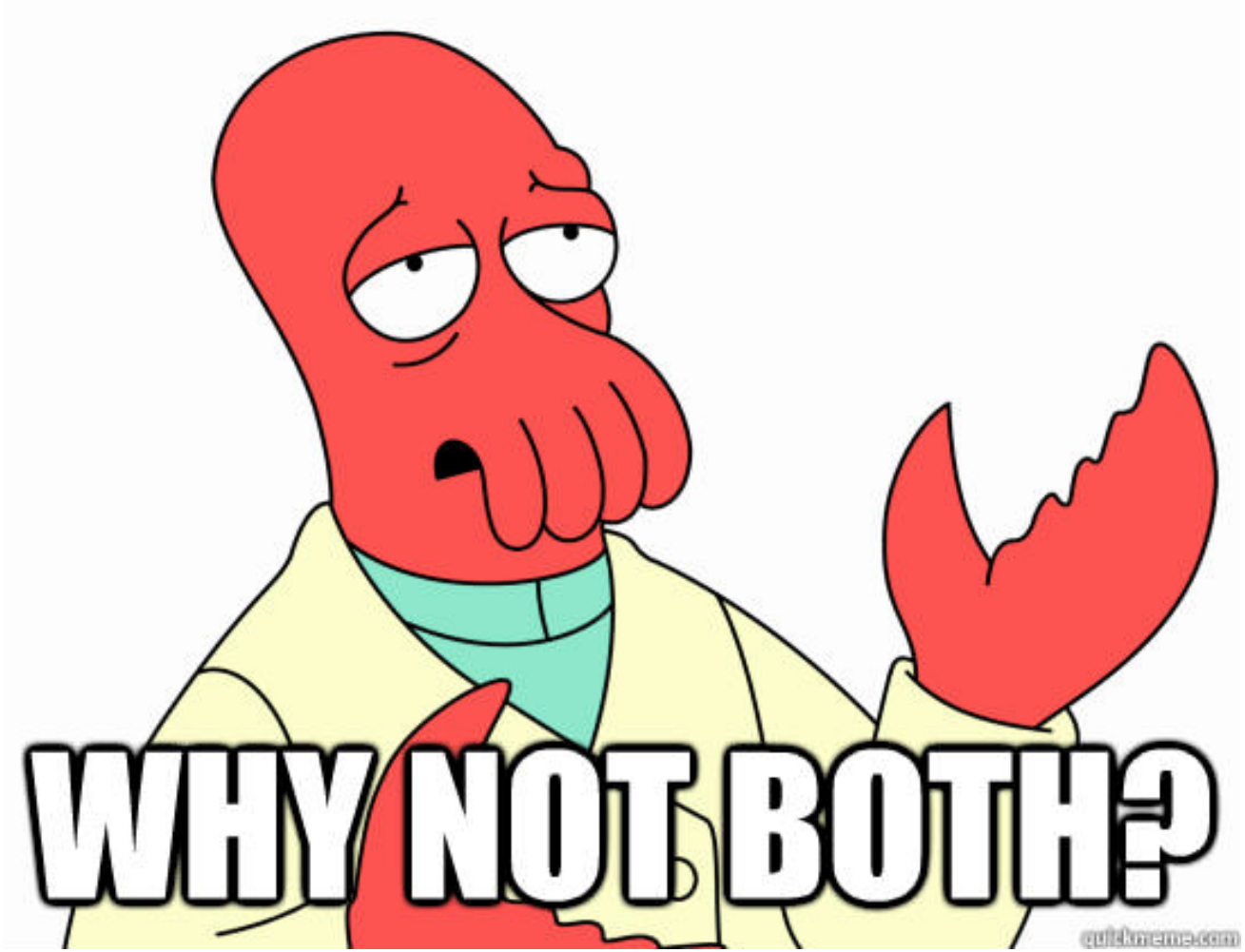
Filter methods

19

Wrapper Methods

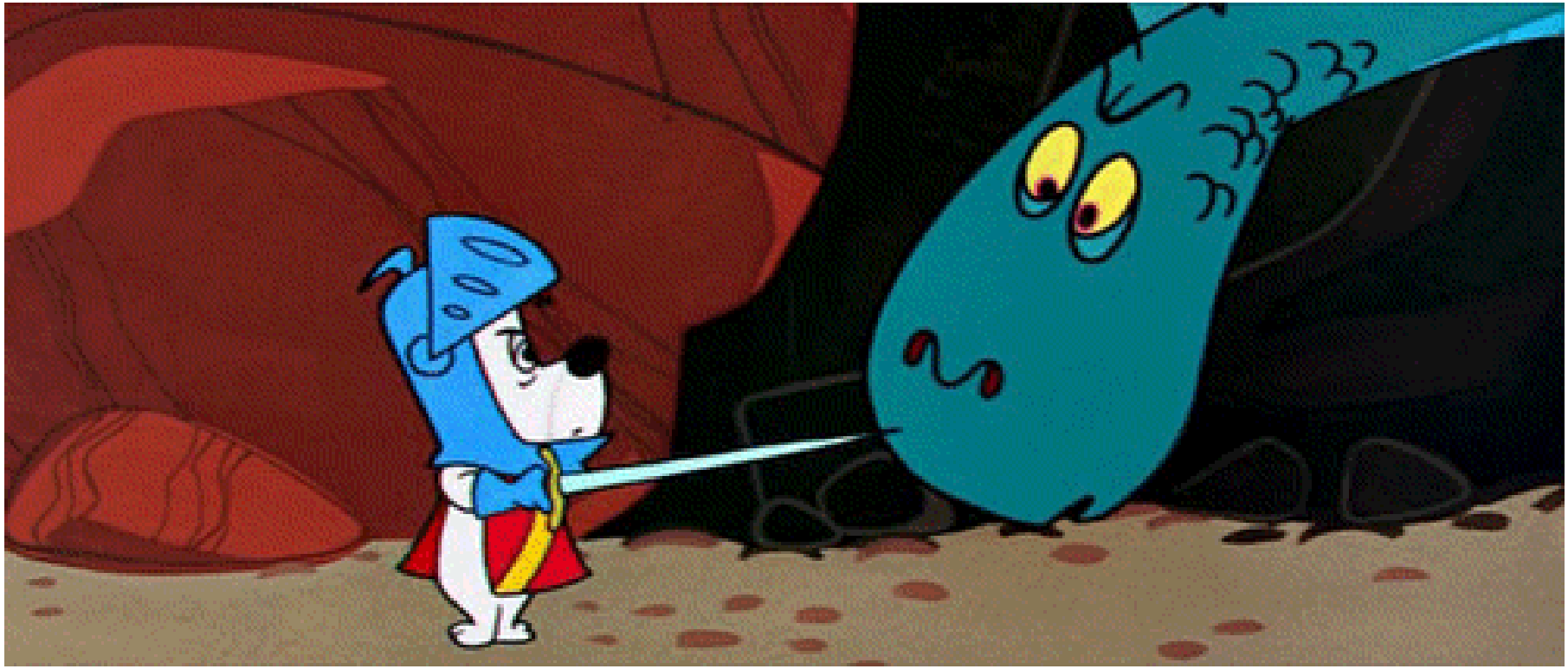
20

Embedded methods



Why are we here again?

22



23

The fight

```
CAVEAT:
The following code is not complete (to promote clarity).
Errors/edge cases handling/docstrings/comments/... are not included.

DO NOT use the code as is: It will NOT work properly...
```

24

The code

We start by importing a bunch of things...

```
import numpy as np
import pandas as pd
from sklearn.base import BaseEstimator

from sklearn.ensemble import RandomForestClassifier, RandomForestRegressor
from sklearn.tree import DecisionTreeClassifier, DecisionTreeRegressor
from sklearn.feature_selection import VarianceThreshold
```

25

We set up our class...

```
class EnsembleFeatureSelector(BaseEstimator):

  __regressors = {
      "RandomForestRegressor": RandomForestRegressor,
      "DecisionTreeRegressor": DecisionTreeRegressor
  }
  __classifiers = {
      "RandomForestClassifier": RandomForestClassifier,
      "DecisionTreeClassifier": DecisionTreeClassifier
  }
  __others = {
      "VarianceThreshold": VarianceThreshold,
  }

  __estimators_by_type = {
      "classification": ["__classifiers"],
      "regression": ["__regressors"],
      "generic_classification": ["__classifiers", "__others"],
      "generic_regression": ["__regressors", "__others"]
  }
```

...and we initialise it

```
def __init__(
        self, number_of_features=-1, analysis_type=None,
        params=None, min_score=0.7, test_split=0
    ):

    ...a bunch of initializations here...
```

27

Models instantiation

```
def _setup_models(self, params):

    estimators = self.__estimators_by_type.get(self.analysis_type)

    for item in estimators:
        for label, est in self.__class__.__dict__[
            f"_{self.__class__.__name__}{item}"
        ].items():
            estimator_kwargs = params.get(label, dict())
            self._active_estimators[label] = est(**estimator_kwargs)
```

28

Fitting

```
    def fit(self, X, y=None):

        number_of_estimators = len(self._active_estimators)

        for n, (key, model) in enumerate(self._active_estimators.items(), start=1):
            model.fit(X, y)

            self._alive_estimators.add(key)
```

29

Feature importance by model

```
@staticmethod
def _calculate_importance(model, names=None):
    if hasattr(model, "feature_importances_"):
        return [(n, i) for n, i in enumerate(model.feature_importances_)]
    if hasattr(model, "variances_"):
        return [(n, i) for n, i in enumerate(model.variances_)]
    if hasattr(model, "scores_"):
        return [(n, i) for n, i in enumerate(model.scores_)]
    return list()


def _get_importances(self):
    for key in self._alive_estimators:
        importances = self._calculate_importance(self._active_estimators[key], self.names)

        self._importances[key] = sorted(
            importances, key=operator.itemgetter(1), reverse=True
        )[:self.n_features]
```

30

Voting

```
    def cast_votes(self, min_votes=0):
        if not self._is_fit:
            raise RuntimeError("Ensemble has not been fit on data. Cannot cast votes")

        self._get_importances()

        votes = collections.Counter(
            feature for feature, _ in itertools.chain(*self._importances.values())
        )
        return collections.Counter(
            {feature: vote for feature, vote in votes.items() if vote >= min_votes}
        )
```

31

Good to go!

Let's see it in action!

33

Remember our dataset?

```
df = pd.DataFrame(data['data'], columns=data['feature_names'])
print(f"Dataset contains {df.shape[0]} rows x {df.shape[1]} columns")
```

Run

Let's run the selector to see what features are important:

```
EFS = EnsembleFeatureSelector(
    analysis_type="generic_classification",
    verbose=1,
    names=df.columns,
    number_of_features=5
)

EFS.fit(df, target)
votes = EFS.cast_votes()

pretty_print_votes(votes)
pretty_print_importances(EFS._importances)
```

Run

34

But does it work?

```
df_train, df_test, target_train, target_test = train_test_split(df, target)

logres.fit(df_train, target_train)
logres_important.fit(df_train[list(votes.keys())], target_train)
```

Run

35

Feature selection:

- simplifies models.
- reduces training time.
- reduces overfitting by enhancing generalization.
- avoids the curse of dimensionality.
- makes data more understandable.
- removes noise.

36

Q/A

Thank you

Pietro Mascolo
Senior Data Scientist - Optum
pietro_mascolo@optum.com (mailto:pietro_mascolo@optum.com)
@iz4vve (http://twitter.com/iz4vve)

Slides and code: bit.ly/2J6xIzj (https://bit.ly/2J6xIzj)