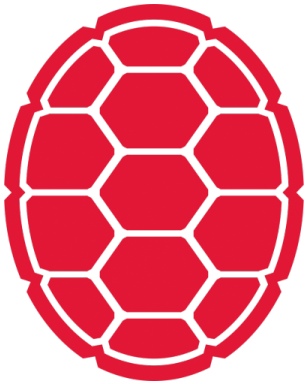

Feature Engineering

Data Science: Jordan Boyd-Graber
University of Maryland
APRIL 5, 2018

Getting Started

```
from csv import DictWriter
from math import log

import pandas as pd
import matplotlib.pyplot as plt
from sklearn.metrics import mean_squared_error
from sklearn import linear_model, feature_extraction

kTARGET = "price2013"
kZIP = [1, 2]
```

Import Data

```
train = pd.read_csv("data/house_train.csv")
test = pd.read_csv("data/house_test.csv")
ans = pd.read_csv("data/house_ans.csv")
```

Create Feature Object, Scale Data

```
# Set up encoder
encoder = feature_extraction.DictVectorizer()
encoder.fit([locations(x) for x in train.iterrows()])

ans["price2013"] /= 1e3
train["price2013"] /= 1e3
```

Location Function

```
def locations(row):
    d = {}
    d["state"] = row[1]["state"]
    # d["county"] = row[1]["county"]
    for ii in kZIP:
    d["zip%i" % ii] = str(row[1]["zip"])[:ii]
    return d
```

Add Features

```
location_df = {}
for dataset, name in [(train, "train"), (test, "test")]:
  location_features = encoder.fit_transform([locations(x) f
                        x in dataset.iterrows()])
  location_df[name] = pd.DataFrame(location_features.toarra
                       columns=encoder.get_feature_names())

  dataset["price2007"] /= 1e6
  dataset["price_squared"] = dataset["price2007"] ** 2
  dataset["missing_pov"] = [1.0 if x < 0 else 0.0
                                  for x in dataset["poverty
  test["poverty"] = [log(2 + x) for x in test["poverty"]]
```

Add Locations

```
train = pd.concat([train, location_df["train"]], axis=1)
test = pd.concat([test, location_df["test"]], axis=1)
```

Create Feature Lists

```
states = [x for x in encoder.get_feature_names() if "state=
zips = [x for x in encoder.get_feature_names() if "="
                            in x and x.startswith("zip")]
features = {"priceonly": ["price2007"],
          #"log": ["price2007", "pricelog"],
          "sqr": ["price2007", "price_squared"],
          "loc": ["price2007", "price_squared"] + states,
          "pov": ["price2007", "price_squared", "poverty", "m
          "zip": ["price2007", "price_squared", "poverty", "m
          }
```

Train Models

```
models = {}
for ii in features:
  mod = linear_model.LinearRegression()
  mod.fit(train[features[ii]], train[["price2013"]])
  models[ii] = mod
```

Create Predictions

```
for ii in models:
  y_train = train[[kTARGET]]
  y_test = ans[[kTARGET]]
  pred_train = models[ii].predict(train[features[ii]])
  pred_test = models[ii].predict(test[features[ii]])
```

Errors

```
for ii in models:
    error_row = {}
    error_row["model"] = ii
    # Missing code ...
    error_row["train"] = mean_squared_error(y_train, pred_tra
    error_row["test"] = mean_squared_error(y_test, pred_test)
    errors.writerow(error_row)
```

Plots

```
for jj in features[ii][:5]:
    fig = plt.figure(figsize=(5, 4))
    fig.suptitle(ii)
    ax = fig.add_subplot(1,1,1)

    ax.scatter(test[[jj]], ans[[kTARGET]] - models[ii].pred
    ax.scatter(train[[jj]], train[[kTARGET]] - models[ii].p

    ax.set_xlabel(jj)
    ax.set_ylabel("Error")

    fig.savefig("%s_%s.png" % (ii, jj))
```