abhayspawar / **featexp**

Feature exploration for supervised learning

#machine-learning #feature-engineering #data-exploration #data-science #visualization

85 commits | **1** branch | **0** packages | **1** release | **2** contributors | MIT

Branch: **master** | New pull request | Create new file | Upload files | Find File | Clone or download

abhayspawar Update README.md — Latest commit `b602d69` on Jul 24

demo/sample_outputs	moved demo notebook	11 months ago
featexp	Comparison between floats is resulting in bug	10 months ago
.gitignore	updated featexp demo notebook	10 months ago
LICENSE	Initial commit	last year
README.md	Update README.md	2 months ago
featexp_demo.ipynb	updated featexp demo notebook	10 months ago
setup.py	updated requirements version	10 months ago

README.md

featexp

Feature exploration for supervised learning. Helps with feature understanding, identifying noisy features, feature debugging, leakage detection and model monitoring.

Installation

```
pip install featexp
```

Using featexp

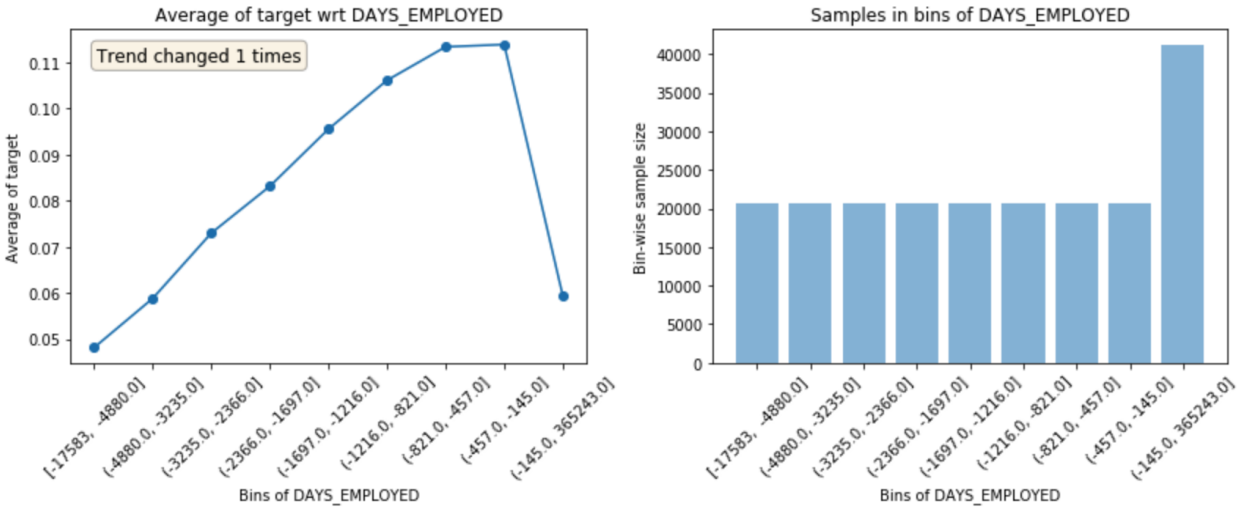
Detailed Medium post on using featexp.

featexp draws plots similar to partial dependence plots, but directly from data instead of using a trained model like current implementations of pdp do. Since, it draws plots from data directly, it helps with understanding the features well and building better ML models.

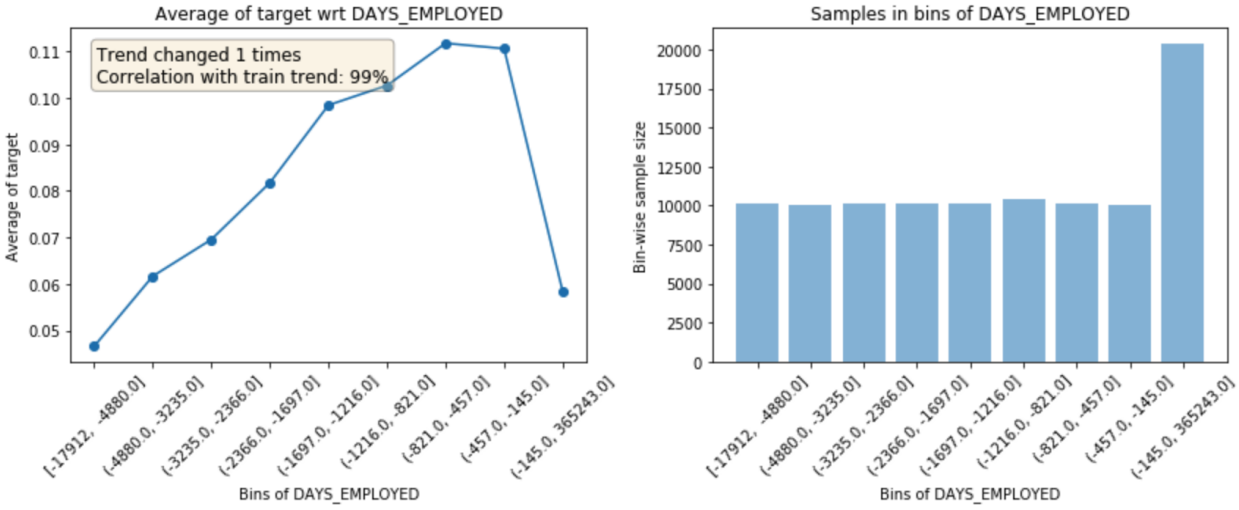
```
from featexp import get_univariate_plots
get_univariate_plots(data=data_train, target_col='target', data_test=data_test, features_list=['DAYS_EMPLOYED'])

# data_test and features_list are optional.
# Draws plots for all columns if features_list not passed
# Draws only train data plots if no test_data passed
```

```
Plots for DAYS_EMPLOYED
Train data plots
```



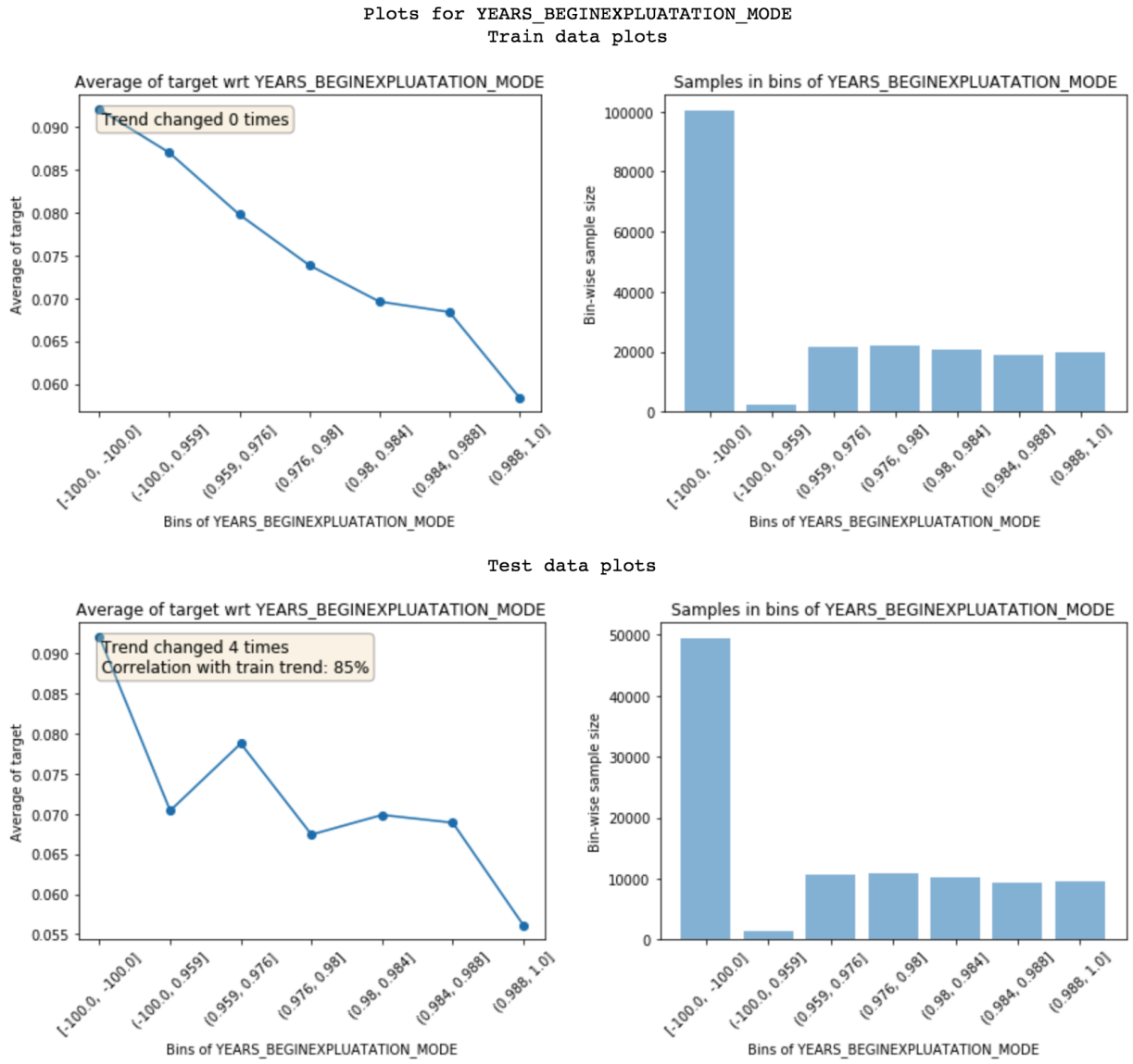

```
Test data plots
```



featexp bins a feature into equal population bins and shows mean value of dependent variable (target) in each bin. Here's how to read these plots:

1. Trend plot on left helps you understand the relationship between target and feature.
2. Population distribution helps you make sure the feature is correct.
3. Also, shows number of trend direction changes and correlation between train and test trend which can be used to identify noisy features. High number of trend changes or low trend correlation implies high noise.

Example of noisy feature: Has low trend correlation



Getting binned feature stats

Returns mean target and population in each bin of a feature

```
from featexp import univariate_plotter
binned_data_train, binned_data_test = univariate_plotter(data=data_train, target_col='target',
feature='DAYS_EMPLOYED', data_test=data_test)
# For only train data
binned_data_train = univariate_plotter(data=data_train, target_col='target', feature='DAYS_EMPLOYED')
```

Getting stats for all features:

Returns trend changes and trend correlation for all features in a dataframe

```
from featexp import get_trend_stats_feature
stats = get_trend_stats(data=data_train, target_col='target', data_test=data_test)

# data_test is optional. If not passed, trend correlations aren't calculated.
```

Returns a dataframe with trend changes and trend correlation which can be used for dropping the noisy features, etc.

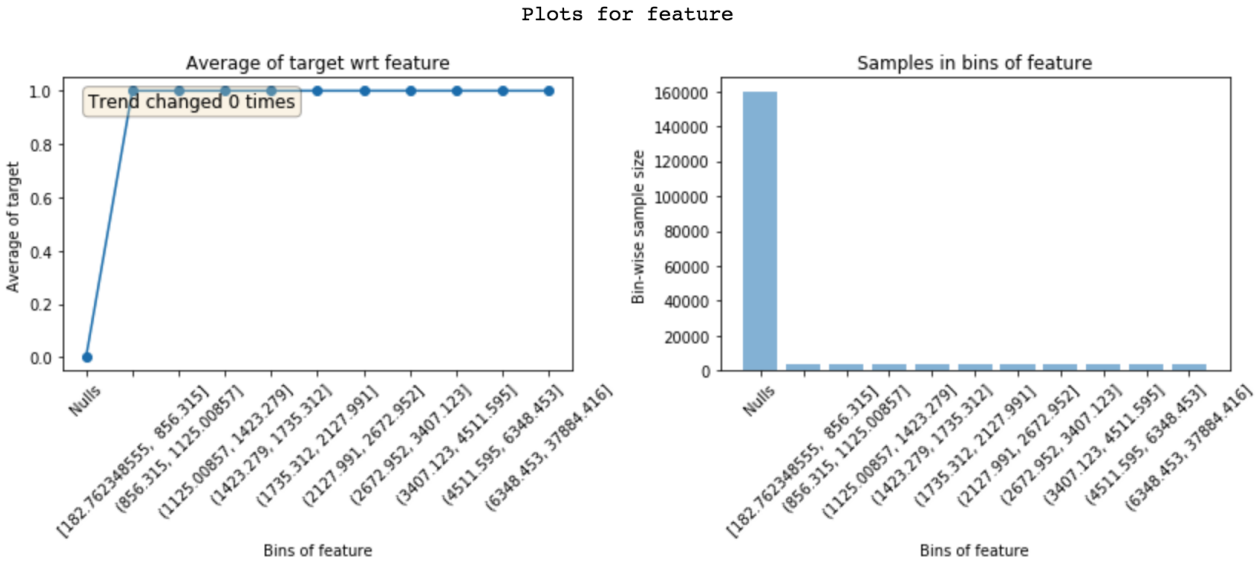
```
In [79]: stats
```

Out[79]:

	Feature	Trend_changes	Trend_changes_test	Trend_correlation
0	CNT_CHILDREN	2	2	0.975688
1	AMT_INCOME_TOTAL	4	3	0.921382
2	AMT_CREDIT	3	3	0.988779
3	AMT_ANNUITY	4	4	0.972325
4	AMT_GOODS_PRICE	7	7	0.994683
5	REGION_POPULATION_RELATIVE	3	3	0.987832
6	DAYS_BIRTH	0	0	0.992783
7	DAYS_EMPLOYED	1	1	0.995426
8	DAYS_REGISTRATION	2	2	0.976891
9	DAYS_ID_PUBLISH	0	2	0.985101
10	OWN_CAR_AGE	1	1	0.994656
11	FLAG_MOBIL	0	0	0.000000

Leakage detection

Helps with identifying why a feature is leaky which helps with debugging.



Nulls have 0% mean target and 100% mean target in other bins. Implies this feature is populated only for target = 1.

Citing featexp

DOI 10.5281/zenodo.2616757