H_2O.ai

Cross-Validated Mean Target Encoding

```python
def mean_target_encoding(data, x, y, fold_column):
    grouped_data = data[[x, fold_column, y]].group_by([x, fold_column])
    grouped_data.sum(na = "ignore").count(na = "ignore")
    df = grouped_data.get_frame().as_data_frame()
    df_list = []
    nfold = int(data[fold_column].max()) + 1
    for j in range(0, nfold):
        te_x = "te_{}".format(x)
        sum_y = "sum_{}".format(y)
        oof = df.loc[df[fold_column] != j, [x, sum_y, "nrow"]]
        stats = oof.groupby([x]).sum()
        stats[x] = stats.index
        stats[fold_column] = j
        stats[te_x] = stats[sum_y] / stats["nrow"]
        df_list.append(stats[[x, fold_column, te_x]])
    return h2o.H2OFrame(pd.concat(df_list))
```

(Feature Engineering Sneak Peak)

Numeric Data Transformations

- h2o_frame[x].abs()
- h2o_frame[x].acos()
- h2o_frame[x].acosh()
- h2o_frame[x].asin()
- h2o_frame[x].asinh()
- h2o_frame[x].atan()
- h2o_frame[x].atanh()
- h2o_frame[x].ceil()
- h2o_frame[x].cos()
- h2o_frame[x].cosh()
- h2o_frame[x].cospi()
- h2o_frame[x].cut(breaks, …)
- h2o_frame[x].digamma()
- h2o_frame[x].exp()
- h2o_frame[x].expm1()
- h2o_frame[x].floor()
- h2o_frame[x].gamma()
- h2o_frame[x].lgamma()
- h2o_frame[x].log()
- h2o_frame[x].log10()
- h2o_frame[x].log1p()
- h2o_frame[x].log2()
- h2o_frame[x].round(digits=0)
- h2o_frame[x].scale(center=True, scale=True)
- h2o_frame[x].sign()
- h2o_frame[x].signif(digits=6)
- h2o_frame[x].sin()
- h2o_frame[x].sinh()
- h2o_frame[x].sinpi()
- h2o_frame[x].sqrt()
- h2o_frame[x].tan()
- h2o_frame[x].tanh()
- h2o_frame[x].tanpi()
- h2o_frame[x].trigamma()
- h2o_frame[x].trunc()

Cross-Validated Mean Target Encoding

(Feature Engineering Sneak Peak)

```python
def mean_target_encoding(data, x, y, fold_column):
    grouped_data = data[[x, fold_column, y]].group_by([x, fold_column])
    grouped_data.sum(na = "ignore").count(na = "ignore")
    df = grouped_data.get_frame().as_data_frame()
    df_list = []
    nfold = int(data[fold_column].max()) + 1
    for j in range(0, nfold):
        te_x = "te_{}".format(x)
        sum_y = "sum_{}".format(y)
        oof = df.loc[df[fold_column] != j, [x, sum_y, "nrow"]]
        stats = oof.groupby([x]).sum()
        stats[x] = stats.index
        stats[fold_column] = j
        stats[te_x] = stats[sum_y] / stats["nrow"]
        df_list.append(stats[[x, fold_column, te_x]])
    return h2o.H2OFrame(pd.concat(df_list))
```