AngieKay / **Presidential-Election-by-County** Public

Code Issues Pull requests Actions Projects Wiki Security

master

Presidential-Election-by-County / README.md

AngieKay Update README.md History

1 contributor

26 lines (23 sloc) | 1.53 KB

Presidential-Election-by-County

Overview

I am building a model to predict presidential elections at the county level.

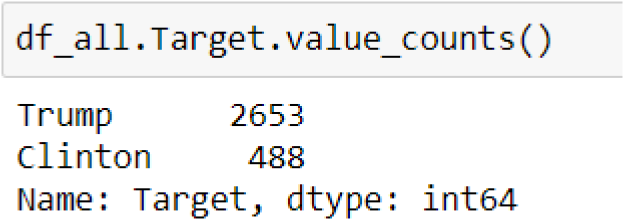
Business Understanding

I am trying to determine the key predictors that lead a county to go either blue or red.

Data & Methodology

2016 and 2012 election results: https://data.world/garyhoov/2016-pres-election-by-county
Alaska 2016 and 2012 election results: https://www.thecinyc.com/
Additional Alaska information:
https://en.wikipedia.org/wiki/List_of_boroughs_and_census_areas_in_Alaska
Race: https://data.census.gov/cedsci/table?q=county%20population&tid=DECENNIALPL2020.P1
Income: https://data.census.gov/cedsci/all?q=county%20population

EDA



```
df_all.Target.value_counts()
```

```
Trump      2653
Clinton     488
Name: Target, dtype: int64
```

Models

All scores are F1.

```
pipe = Pipeline([
    ('scaler', PipelineHelper([
        ('std', StandardScaler()),
        ('max', MaxAbsScaler()),
        ('minmax', MinMaxScaler())
    ])),
    ('classifier', PipelineHelper([
        ('svm', LinearSVC()),
        ('rf', RandomForestClassifier()),
        ('logreg', LogisticRegression()),
        ('dt', DecisionTreeClassifier())
    ])),
])
```

```
params = {
    'scaler__selected_model': pipe.named_steps['scaler'].generate({
        'std__with_mean': [True, False],
        'std__with_std': [True, False],
        'max__copy': [True],  # just for displaying
    }),
    'classifier__selected_model': pipe.named_steps['classifier'].generate({
#          'svm__C': [None, 1.0],
#          'svm__kernel': ['rbf', 'poly', 'linear'],
#          'svm__penalty': ['l1', 'l2'],
        'svm__class_weight': [None, 'balanced'],
        'rf__max_depth': [None, 5, 10, 30],
        'rf__class_weight': [None, 'balanced'],
        'rf__n_estimators': [100, 20],
        'logreg__penalty': [None, 'l1', 'l2', 'elasticnet'],
        'logreg__C': [0.1, 1.0],
        'logreg__class_weight': [None, 'balanced'],
        'logreg__solver': ['lbfgs', 'liblinear', 'sag', 'saga'],
        'dt__class_weight': [None, 'balanced']
    })
}
```

```
grid.best_score_
```

```
0.8679769922229997
```

```
grid.best_params_
```

```
{'classifier__selected_model': ('logreg',
  {'C': 0.1, 'class_weight': 'balanced', 'penalty': 'l2', 'solver': 'lbfgs'}),
 'scaler__selected_model': ('std', {'with_mean': False, 'with_std': False})}
```

```
confusion_matrix(y_train, y_hat_train)
```

```
array([[1967,   22],
       [  53,  313]], dtype=int64)
```

Final Model For Now

All scores are F1.

```
pipeline = imbpipe(steps = [
    ('sm', SMOTE()),
    ('ss', StandardScaler(with_mean = False, with_std = False)),
    ('linsvc', LinearSVC(class_weight = 'balanced'))
])
```

```
results['test_score'].mean()
```

```
0.8756135291732597
```

```
results['train_score'].mean()
```

```
0.8785950018179409
```

```
confusion_matrix(y_train, y_hat_train)
```

```
array([[1909,   80],
       [  19,  347]], dtype=int64)
```