EXTREME GRADIENT BOOSTING WITH XGBOOST

Review of pipelines using sklearn

Sergey Fogelson
VP of Analytics, Viacom

Pipeline Review

- Takes a list of named 2-tuples (name, pipeline_step) as input
- Tuples can contain any arbitrary scikit-learn compatible estimator or transformer object
- Pipeline implements fit/predict methods
- Can be used as input estimator into grid/randomized search and cross_val_score methods

Scikit-learn pipeline example

```
In [1]: import pandas as pd
   ...: from sklearn.ensemble import RandomForestRegressor
   ...: import numpy as np
   ...: from sklearn.preprocessing import StandardScaler
   ...: from sklearn.pipeline import Pipeline
   ...: from sklearn.model_selection import cross_val_score

In [2]: names = ["crime","zone","industry","charles",
   ...: "no","rooms","age", "distance",
   ...: "radial","tax","pupil","aam","lower","med_price"]

In [3]: data = pd.read_csv("boston_housing.csv",names=names)

In [4]: X, y = data.iloc[:,:-1], data.iloc[:,-1]

In [5]: rf_pipeline = Pipeline[("st_scaler",
   ...: StandardScaler()),
   ...: ("rf_model",RandomForestRegressor())]

In [6]: scores = cross_val_score(rf_pipeline,X,y,
```

Scikit-learn pipeline example

```
In [7]: final_avg_rmse = np.mean(np.sqrt(np.abs(scores)))

In [8]: print("Final RMSE:", final_avg_rmse)
Final RMSE: 4.54530686529
```

Preprocessing I: LabelEncoder and OneHotEncode

- LabelEncoder: Converts a categorical column of strings into integers
- OneHotEncoder: Takes the column of integers and encodes them as dummy variables
- Cannot be done within a pipeline

Preprocessing II: DictVectorizer

- Traditionally used in text processing
- Converts lists of feature mappings into vectors
- Need to convert DataFrame into a list of dictionary entries
- Explore the scikit-learn documentation

EXTREME GRADIENT BOOSTING WITH XGBOOST

Let's build pipelines!

EXTREME GRADIENT BOOSTING WITH XGBOOST

Incorporating xgboost into pipelines

Sergey Fogelson

Scikit-Learn Pipeline Example With XGBoost

```
In [1]: import pandas as pd
   ...: import xgboost as xgb
   ...: import numpy as np
   ...: from sklearn.preprocessing import StandardScaler
   ...: from sklearn.pipeline import Pipeline
   ...: from sklearn.model_selection import cross_val_score

In [2]: names = ["crime","zone","industry","charles","no",
   ...: "rooms","age", "distance","radial","tax",
   ...: "pupil","aam","lower","med_price"]
In [3]: data = pd.read_csv("boston_housing.csv",names=names)
In [4]: X, y = data.iloc[:,:-1], data.iloc[:,-1]

In [5]: xgb_pipeline = Pipeline[("st_scaler",
   ...: StandardScaler()),
   ...: ("xgb_model",xgb.XGBRegressor())]
In [6]: scores = cross_val_score(xgb_pipeline, X, y,
   ...:scoring="neg_mean_squared_error",cv=10)

In [7]: final_avg_rmse = np.mean(np.sqrt(np.abs(scores)))
```

Additional Components Introduced For Pipelines

- sklearn_pandas:
  - DataFrameMapper - Interoperability between pandas and scikit-learn
  - CategoricalImputer - Allow for imputation of categorical variables before conversion to integers
- sklearn.preprocessing:
  - Imputer - Native imputation of numerical columns in scikit-learn
- sklearn.pipeline:
  - FeatureUnion - combine multiple pipelines of features into a single

EXTREME GRADIENT BOOSTING WITH XGBOOST

Let's practice!

EXTREME GRADIENT BOOSTING WITH XGBOOST

Tuning xgboost hyperparameters in a pipeline

Sergey Fogelson
VP of Analytics, Viacom

Tuning XGBoost hyperparameters in a Pipeline

```
In [1]: import pandas as pd
   ...: import xgboost as xgb
   ...: import numpy as np
   ...: from sklearn.preprocessing import StandardScaler
   ...: from sklearn.pipeline import Pipeline
   ...: from sklearn.model_selection import RandomizedSearchCV

In [2]: names = ["crime","zone","industry","charles","no",
   ...: "rooms","age", "distance","radial","tax",
   ...: "pupil","aam","lower","med_price"]
In [3]: data = pd.read_csv("boston_housing.csv",names=names)
In [4]: X, y = data.iloc[:,:-1],data.iloc[:,-1]
In [5]: xgb_pipeline = Pipeline[("st_scaler",
   ...: StandardScaler()), ("xgb_model",xgb.XGBRegressor())]

In [6]: gbm_param_grid = {
   ...:     'xgb_model__subsample': np.arange(.05, 1, .05),
   ...:     'xgb_model__max_depth': np.arange(3,20,1),
   ...:     'xgb_model__colsample_bytree': np.arange(.1,1.05,.05) }
```

Tuning XGBoost hyperparameters in a Pipeline II

```
In [9]: print("Best rmse: ",
   ...: np.sqrt(np.abs(randomized_neg_mse.best_score_)))
Best rmse: 3.9966784203040677

In [10]: print("Best model: ",
    ...: randomized_neg_mse.best_estimator_)
Best model:  Pipeline(steps=[('st_scaler', StandardScaler(copy=True,
with_mean=True, with_std=True)),
('xgb_model', XGBRegressor(base_score=0.5, colsample_bylevel=1,
       colsample_bytree=0.95000000000000029, gamma=0, learning_rate=0.1,
       max_delta_step=0, max_depth=8, min_child_weight=1, missing=None,
       n_estimators=100, nthread=-1, objective='reg:linear', reg_alpha=0,
       reg_lambda=1, scale_pos_weight=1, seed=0, silent=True,
       subsample=0.90000000000000013))])
```

EXTREME GRADIENT BOOSTING WITH XGBOOST

Let's finish this up!

EXTREME GRADIENT BOOSTING WITH XGBOOST

Final Thoughts

Sergey Fogelson
VP of Analytics, Viacom

What We Have Covered And You Have Learned

- Using XGBoost for classification tasks
- Using XGBoost for regression tasks
- Tuning XGBoost's most important hyperparameters
- Incorporating XGBoost into sklearn pipelines

What We Have Not Covered (And How You Can Pro

- Using XGBoost for ranking/recommendation problems (Netflix/Amazon problem)
- Using more sophisticated hyperparameter tuning strategies for tuning XGBoost models (Bayesian Optimization)
- Using XGBoost as part of an ensemble of other models for regression/classification

EXTREME GRADIENT BOOSTING WITH XGBOOST

Congratulations!