Review of pipelines using sklearn

EXTREME GRADIENT BOOSTING WITH XGBOOST

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

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

data = pd.read_csv("boston_housing.csv",names=names)

X, y = data.iloc[:,:-1], data.iloc[:,-1]
rf_pipeline = Pipeline[("st_scaler",
                StandardScaler()),
                ("rf_model",RandomForestRegressor())]

scores = cross_val_score(rf_pipeline,X,y,
scoring="neg_mean_squared_error",cv=10)
```

Scikit-learn pipeline example

```
final_avg_rmse = np.mean(np.sqrt(np.abs(scores)))

print("Final RMSE:", final_avg_rmse)
```

```
Final RMSE: 4.54530686529
```

Preprocessing I: LabelEncoder and OneHotEncoder

- `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**

Let's build pipelines!

EXTREME GRADIENT BOOSTING WITH XGBOOST

Incorporating xgboost into pipelines

EXTREME GRADIENT BOOSTING WITH XGBOOST

Sergey Fogelson
VP of Analytics, Viacom

Scikit-learn pipeline example with XGBoost

```
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
names = ["crime","zone","industry","charles","no","rooms","age",
        "distance","radial","tax","pupil","aam","lower","med_price"]
data = pd.read_csv("boston_housing.csv",names=names)
X, y = data.iloc[:,:-1], data.iloc[:,-1]
xgb_pipeline = Pipeline[("st_scaler", StandardScaler()),
                        ("xgb_model",xgb.XGBRegressor())]
scores = cross_val_score(xgb_pipeline, X, y,
                         scoring="neg_mean_squared_error",cv=10)
final_avg_rmse = np.mean(np.sqrt(np.abs(scores)))
print("Final XGB RMSE:", final_avg_rmse)
```

```
Final RMSE: 4.02719593323
```

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 pipeline of features

Let's practice!

EXTREME GRADIENT BOOSTING WITH XGBOOST

Tuning xgboost hyperparameters in a pipeline

EXTREME GRADIENT BOOSTING WITH XGBOOST

Sergey Fogelson
VP of Analytics, Viacom

Tuning XGBoost hyperparameters in a pipeline

```
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
names = ["crime","zone","industry","charles","no",
   ...: "rooms","age", "distance","radial","tax",
   ...: "pupil","aam","lower","med_price"]
data = pd.read_csv("boston_housing.csv",names=names)
X, y = data.iloc[:,:-1],data.iloc[:,-1]
xgb_pipeline = Pipeline[("st_scaler",
   ...: StandardScaler()), ("xgb_model",xgb.XGBRegressor())]
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) }
randomized_neg_mse = RandomizedSearchCV(estimator=xgb_pipeline,
   ...: param_distributions=gbm_param_grid, n_iter=10,
   ...: scoring='neg_mean_squared_error', cv=4)
randomized_neg_mse.fit(X, y)
```

Tuning XGBoost hyperparameters in a pipeline II

```
print("Best rmse: ", np.sqrt(np.abs(randomized_neg_mse.best_score_)))
```

```
Best rmse: 3.9966784203040677
```

```
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))])
```

Let's finish this up!

EXTREME GRADIENT BOOSTING WITH XGBOOST

Final Thoughts

EXTREME GRADIENT BOOSTING WITH XGBOOST

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 Proceed)

- 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

Congratulations!

EXTREME GRADIENT BOOSTING WITH XGBOOST