7506R_TP2_GRUPO16_ENTREGA_XGBooost

December 8, 2022

[1]:
```
!pip install xgboost==1.6.2
import pandas as pd
import numpy as np
import statistics
import xgboost as xgb
from sklearn.model_selection import RandomizedSearchCV, GridSearchCV,␣
 ↪cross_val_score
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import confusion_matrix, precision_recall_curve,␣
 ↪roc_curve, recall_score, accuracy_score,f1_score, precision_score, auc,␣
 ↪roc_auc_score, mean_squared_error, silhouette_score,␣
 ↪classification_report,plot_roc_curve,mean_absolute_error, max_error,␣
 ↪median_absolute_error, r2_score, explained_variance_score
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-
wheels/public/simple/
Collecting xgboost==1.6.2
  Downloading xgboost-1.6.2-py3-none-manylinux2014_x86_64.whl (255.9 MB)
     |                  | 255.9 MB 27 kB/s
Requirement already satisfied: numpy in /usr/local/lib/python3.8/dist-
packages (from xgboost==1.6.2) (1.21.6)
Requirement already satisfied: scipy in /usr/local/lib/python3.8/dist-packages
(from xgboost==1.6.2) (1.7.3)
Installing collected packages: xgboost
  Attempting uninstall: xgboost
    Found existing installation: xgboost 0.90
    Uninstalling xgboost-0.90:
      Successfully uninstalled xgboost-0.90
Successfully installed xgboost-1.6.2
```

[2]:
```
url = 'https://drive.google.com/file/d/1Sh8wEwQNuAqXGkKEYwAWm97I0r6rAm64/view?
 ↪usp=share_link'
path = 'https://drive.google.com/uc?export=download&id='+url.split('/')[-2]

ds_train = pd.read_csv(path)
```

```
url = 'https://drive.google.com/file/d/1Zrzwx7Aw9Ws9ohvbZOkl7lesL4Klzb67/view?
 ↪usp=share_link'
path = 'https://drive.google.com/uc?export=download&id='+url.split('/')[-2]

ds_test = pd.read_csv(path)
```

[3]:
```
de_temp_train = ds_train.copy()

de_temp_train.loc[ds_train["tipo_precio"]=="bajo", "target"]= 0
de_temp_train.loc[ds_train["tipo_precio"]=="medio", "target"]= 1
de_temp_train.loc[ds_train["tipo_precio"]=="alto", "target"]= 2
```

[4]:
```
de_temp_test = ds_test.copy()

de_temp_test.loc[ds_test["tipo_precio"]=="bajo", "target"]= 0
de_temp_test.loc[ds_test["tipo_precio"]=="medio", "target"]= 1
de_temp_test.loc[ds_test["tipo_precio"]=="alto", "target"]= 2
```

[5]:
```
ds_train_x = de_temp_train.drop(['id', 'tipo_precio', 'property_price',␣
 ↪'target'], axis='columns', inplace=False)
ds_test_x = de_temp_test.drop(['id', 'tipo_precio', 'property_price',␣
 ↪'target'], axis='columns', inplace=False)
```

[6]:
```
ds_train_x = pd.get_dummies(ds_train_x, columns=["barrio","property_type"],␣
 ↪drop_first=True)
ds_test_x = pd.get_dummies(ds_test_x, columns=["barrio","property_type"],␣
 ↪drop_first=True)
```

[7]:
```
ds_train_y = de_temp_train['property_price'].copy()
ds_test_y = de_temp_test['property_price'].copy()
```

[8]:
```
def report_regression(model, x_train, y_train, x_test, y_test):
    # Get a prediction over x_test
    y_pred = model.predict(x_test)

    error = y_test - y_pred
    percentil = [25,50,75]
    percentil_value = np.percentile(error, percentil)

    metrics = [
        ('mean absolute error', mean_absolute_error(y_test, y_pred)),
        ('median absolute error', median_absolute_error(y_test, y_pred)),
        ('mean squared error', mean_squared_error(y_test, y_pred)),
        ('max error', max_error(y_test, y_pred)),
        ('r2 score', r2_score(y_test, y_pred)),
        ('explained variance score', explained_variance_score(y_test, y_pred))
    ]
```

```
    print('Metricas for regression:')
    for metric_name, metric_value in metrics:
        print(f'{metric_name:>25s}: {metric_value: >20.3f}')

    print('\nPercentiles:')
    for p, pv in zip(percentil, percentil_value):
        print(f'{p: 25d}: {pv:>20.3f}')

    # Calculate various precision and accuracy score
    score = model.score(x_test, y_test)
    cv_score = cross_val_score(model, x_train, y_train, cv=5)


    # Print all scores
    print(f"El score general del modelo es {score}")
    print(
        f"La media del cross validation score con k=5 es {statistics.
↪median(cv_score)}"
    )
```

0.0.1 XGBoost

Utilizamos los hiperparametros obtenidos en el tp1

[9]:
```
modelo_tp1 = xgb.XGBRegressor(base_score=0.5, booster='gbtree', callbacks=None,
             colsample_bylevel=1, colsample_bynode=1, colsample_bytree=1,
             early_stopping_rounds=None, enable_categorical=False,
             eval_metric='rmse', gamma=0, gpu_id=-1, grow_policy='depthwise',
             importance_type=None, interaction_constraints='',
             learning_rate=0.3, max_bin=256, max_cat_to_onehot=4,
             max_delta_step=0, max_depth=6, max_leaves=0, min_child_weight=1,
             monotone_constraints='()', n_estimators=100, n_jobs=0,
             num_parallel_tree=1, predictor='auto', random_state=0, reg_alpha=0,
             reg_lambda=1)
```

[10]:
```
modelo_tp1.fit(ds_train_x, ds_train_y)
```

[10]:
```
XGBRegressor(base_score=0.5, booster='gbtree', callbacks=None,
             colsample_bylevel=1, colsample_bynode=1, colsample_bytree=1,
             early_stopping_rounds=None, enable_categorical=False,
             eval_metric='rmse', gamma=0, gpu_id=-1, grow_policy='depthwise',
             importance_type=None, interaction_constraints='',
             learning_rate=0.3, max_bin=256, max_cat_to_onehot=4,
             max_delta_step=0, max_depth=6, max_leaves=0, min_child_weight=1,
             missing=nan, monotone_constraints='()', n_estimators=100, n_jobs=0,
             num_parallel_tree=1, predictor='auto', random_state=0, reg_alpha=0,
```

```
                reg_lambda=1, …)
```

[11]:
```
report_regression(modelo_tp1, ds_train_x, ds_train_y, ds_test_x, ds_test_y)
```

```
Metricas for regression:
      mean absolute error:            40077.612
    median absolute error:            18819.500
       mean squared error:       8504629256.109
                max error:          2487949.250
                 r2 score:                0.894
 explained variance score:                0.895

Percentiles:
                       25:           -13539.340
                       50:             3696.492
                       75:            24972.504
El score general del modelo es 0.8943279620167106
La media del cross validation score con k=5 es 0.892281384846324
```

Optimizamos hiperparametros con el nuevo data set

[12]:
```
parameters = { 'objective' : ['reg:squarederror', 'reg:squaredlogerror'],␣
 ↪'n_estimators' : [100], 'max_depth' : [6], 'learning_rate' : [0.3, 0.5],␣
 ↪'booster': ['gbtree', 'dart'], 'eval_metric' : ['rmse'] }
model = xgb.XGBRegressor()
```

[13]:
```
gscv = RandomizedSearchCV(model, parameters,␣
 ↪scoring='neg_root_mean_squared_error', n_jobs=-1, refit=True)

gscv.fit(ds_train_x, ds_train_y) #20 minutos
```

```
/usr/local/lib/python3.8/dist-packages/sklearn/model_selection/_search.py:292:
UserWarning: The total space of parameters 8 is smaller than n_iter=10. Running
8 iterations. For exhaustive searches, use GridSearchCV.
  warnings.warn(
```

[13]:
```
RandomizedSearchCV(estimator=XGBRegressor(base_score=None, booster=None,
                                          callbacks=None,
                                          colsample_bylevel=None,
                                          colsample_bynode=None,
                                          colsample_bytree=None,
                                          early_stopping_rounds=None,
                                          enable_categorical=False,
                                          eval_metric=None, gamma=None,
                                          gpu_id=None, grow_policy=None,
                                          importance_type=None,
                                          interaction_constraints=None,
                                          learning_rate=None, max_bin=None,
```

```
                                    max_ca…
                                    n_estimators=100, n_jobs=None,
                                    num_parallel_tree=None,
                                    predictor=None, random_state=None,
                                    reg_alpha=None, reg_lambda=None, …),
                   n_jobs=-1,
                   param_distributions={'booster': ['gbtree', 'dart'],
                                        'eval_metric': ['rmse'],
                                        'learning_rate': [0.3, 0.5],
                                        'max_depth': [6], 'n_estimators': [100],
                                        'objective': ['reg:squarederror',
                                                      'reg:squaredlogerror']},
                   scoring='neg_root_mean_squared_error')
```

[14]:
```
report_regression(gscv.best_estimator_, ds_train_x, ds_train_y, ds_test_x,␣
↪ds_test_y)
```

```
Metricas for regression:
      mean absolute error:            40077.612
    median absolute error:            18819.500
       mean squared error:       8504629256.109
                max error:          2487949.250
                 r2 score:                0.894
 explained variance score:                0.895

Percentiles:
                      25:           -13539.340
                      50:             3696.492
                      75:            24972.504
El score general del modelo es 0.8943279620167106
La media del cross validation score con k=5 es 0.892281384846324
```

[15]:
```
# Guardamos el modelo
import pickle

filename = 'xgboost_tp2.sav'
pickle.dump(gscv, open(filename, 'wb'))
```

Concluciones

Realizamos las predicciones sobre el nuevo set de datos utilizando los hiperparametros obtenidos en el primer trabajo practico y los obtenidos en este segundo trabajo. En ambos casos la busqueda de hiperparametros arrojo los mismos resultados por lo que ambos modelos resultan ser identicos y como concecuencia el score es el mismo en ambos.

En comparacion con los resutlados obtenidos en el trabajo anterior con el set de datos sin la informacion de las descripciones el score se redujo minimamente, pasamos de un score general de 0,90 a un score general de 0,892. Por lo que podemos decir que la nueva informacion agregada al data set no nos fue de utilidad.