Paris Housing & Swimming Pools

Tyler Cambron

Paris Housing Data

I am trying to find out whether or not having a pool is correlated with the rest of these variables.

The column “hasPool” was chosen as my “y”, leaving the rest as “X”.

Variable “category” turned into “category_Basic” and “category_Luxury” using dummy variables.

#	Column	Non-Null Count	Dtype
0	squareMeters	10000 non-null	int64
1	numberOfRooms	10000 non-null	int64
2	hasYard	10000 non-null	int64
3	hasPool	10000 non-null	int64
4	floors	10000 non-null	int64
5	cityCode	10000 non-null	int64
6	cityPartRange	10000 non-null	int64
7	numPrevOwners	10000 non-null	int64
8	made	10000 non-null	int64
9	isNewBuilt	10000 non-null	int64
10	hasStormProtector	10000 non-null	int64
11	basement	10000 non-null	int64
12	attic	10000 non-null	int64
13	garage	10000 non-null	int64
14	hasStorageRoom	10000 non-null	int64
15	hasGuestRoom	10000 non-null	int64
16	price	10000 non-null	float64
17	category	10000 non-null	object

Random Search CV

Using random search cross-validation, I was able to the four different models: logistic regression, support vector regression, decisions trees, and random forest.

Each model has its own set of parameters. I created tables to test for the best possible parameters for each model.

Each model was saved to a dictionary, and then that dictionary looped to fit all of the models and find predictions.

```
from sklearn.linear_model import LogisticRegression
logistic_params = {
    'tol': [1e-3, 1e-4, 1e-5],
    'solver': ['liblinear'],
    'C': [0.1, 1, 10, 100],
    'n_jobs': [1,2,3,4,5],
    'random_state': [0],
}
rscv['logistic'] = {
    'label': 'Logistic Regressor',
    'model': RandomizedSearchCV(LogisticRegression(), logistic_params, verbose=3, n_iter=3)
}
```

```
for clf in rscv:
    rscv[clf]['model'].fit(X_train, y_train)
    y_pred = rscv[clf]['model'].predict(X_test)
    table.add_row([rscv[clf]['label'], rscv[clf]['model'].best_score_,
                   rscv[clf]['model'].best_params_,
                   mean_squared_error(y_test, y_pred, squared=False)])
```

Results

The results of each of the models were put into a PrettyTable.

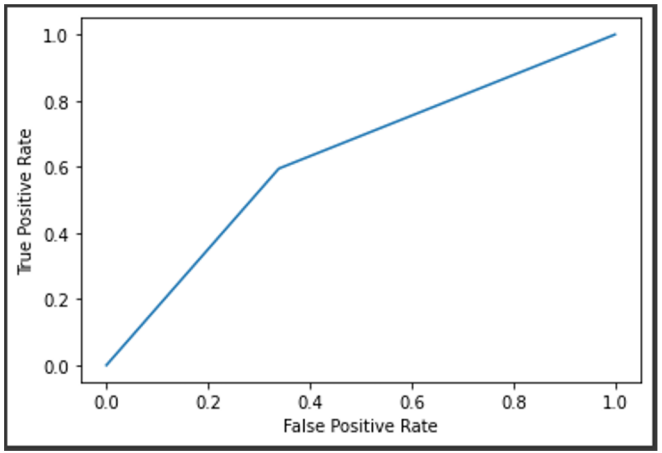
The results contain the model's best score, best parameters, and the calculation of the root mean squared error (RMSE) based off of a predicted value of y (y_pred).

Model	Best Score	Best Params	RMSE
Logistic Regressor	0.6241176470588237	{'tol': 0.0001, 'solver': 'liblinear', 'random_state': 0, 'n_jobs': 5, 'C': 0.1}	0.6186005711819758
C-Support Vector Classification	0.6310588235294118	{'tol': 0.001, 'random_state': 0, 'degree': 1, 'C': 0.1}	0.6148170459575759
Decision Tree Classifier	0.628	{'random_state': 0, 'max_depth': 15, 'criterion': 'gini'}	0.6099180272790763
Random Forest Classifier	0.6309411764705883	{'random_state': 0, 'n_estimators': 10, 'max_depth': 10, 'criterion': 'gini'}	0.6164414002968976

The lowest RMSE is decision tree classifier. So assumably it is the best model for this data.

Results

ROC Curve calculated with a prediction of y from the decision tree classifier:



The area under the curve score is 0.6278.

Classification Report:

	precision	recall	f1-score	support
0	0.62	0.66	0.64	755
1	0.63	0.59	0.61	745
accuracy			0.63	1500
macro avg	0.63	0.63	0.63	1500
weighted avg	0.63	0.63	0.63	1500