Day_16_Ensemble_Methods_for_Regression

November 13, 2023

1 Ensemble Methods for Regression

In these exercises, use scikit-learn whenever possible!

[1]:
```
import pandas as pd

import numpy as np

from sklearn.preprocessing import StandardScaler
from sklearn.neighbors import KNeighborsRegressor
from sklearn.linear_model import LinearRegression
from sklearn.pipeline import make_pipeline
from sklearn.model_selection import cross_val_score
from sklearn.ensemble import VotingRegressor
from sklearn.ensemble import StackingRegressor
```

1.1 Ames Housing Data

In the Day 15 notebook, you tried to find a single model for predicting SalePrice that resulted in your lowest test RMSE; the model is determined by its features, method (linear or kNN), value of k if applicable, etc.

Now you'll work with a partner to create an ensemble model.

Note: If you and your partner worked together to come up with the same model from Day 15, you'll need to find a partner with a different model!

[2]:
```
df_ames = pd.read_csv("https://raw.githubusercontent.com/kevindavisross/data301/
    ↪main/data/AmesHousing.txt", sep="\t")
df_ames
```

[2]:
```
      Order        PID  MS SubClass MS Zoning  Lot Frontage  Lot Area Street
0         1  526301100           20        RL         141.0     31770   Pave  \
1         2  526350040           20        RH          80.0     11622   Pave
2         3  526351010           20        RL          81.0     14267   Pave
3         4  526353030           20        RL          93.0     11160   Pave
4         5  527105010           60        RL          74.0     13830   Pave
…      …        …           …         …                    …        …  …
2925   2926  923275080           80        RL          37.0      7937   Pave
```

```
2926    2927  923276100          20        RL         NaN     8885   Pave
2927    2928  923400125          85        RL        62.0    10441   Pave
2928    2929  924100070          20        RL        77.0    10010   Pave
2929    2930  924151050          60        RL        74.0     9627   Pave

     Alley Lot Shape Land Contour  … Pool Area Pool QC  Fence Misc Feature
0      NaN       IR1          Lvl  …         0     NaN    NaN          NaN  \
1      NaN       Reg          Lvl  …         0     NaN  MnPrv          NaN
2      NaN       IR1          Lvl  …         0     NaN    NaN         Gar2
3      NaN       Reg          Lvl  …         0     NaN    NaN          NaN
4      NaN       IR1          Lvl  …         0     NaN  MnPrv          NaN
…      …         …          …   …        …      …     …            …
2925   NaN       IR1          Lvl  …         0     NaN  GdPrv          NaN
2926   NaN       IR1          Low  …         0     NaN  MnPrv          NaN
2927   NaN       Reg          Lvl  …         0     NaN  MnPrv         Shed
2928   NaN       Reg          Lvl  …         0     NaN    NaN          NaN
2929   NaN       Reg          Lvl  …         0     NaN    NaN          NaN

     Misc Val Mo Sold Yr Sold Sale Type  Sale Condition  SalePrice
0           0       5    2010       WD           Normal     215000
1           0       6    2010       WD           Normal     105000
2       12500       6    2010       WD           Normal     172000
3           0       4    2010       WD           Normal     244000
4           0       3    2010       WD           Normal     189900
…         …     …     …        …            …         …
2925        0       3    2006       WD           Normal     142500
2926        0       6    2006       WD           Normal     131000
2927      700       7    2006       WD           Normal     132000
2928        0       4    2006       WD           Normal     170000
2929        0      11    2006       WD           Normal     188000

[2930 rows x 82 columns]
```

[3]:
```
df_ames_train = df_ames.loc[:1465].copy()
df_ames_test = df_ames.loc[1466:].copy()
x_temp = df_ames_train[["Lot Area", "Pool Area", "Total Bsmt SF", "TotRms AbvGrd", "Garage Cars"]]
X_train = x_temp.fillna(x_temp.mean())
y_train = df_ames_train["SalePrice"]

x2_temp = df_ames_test[["Lot Area", "Pool Area", "Total Bsmt SF", "TotRms AbvGrd", "Garage Cars"]]
X_test = x_temp.fillna(x2_temp.mean())
```

[4]:
```
knn_model = make_pipeline(
    StandardScaler(),
    KNeighborsRegressor(n_neighbors=8,
```

```
                        metric="manhattan")
)
knn_model.fit(X_train, y_train)
```

[4]:
```
Pipeline(steps=[('standardscaler', StandardScaler()),
                ('kneighborsregressor',
                 KNeighborsRegressor(metric='manhattan', n_neighbors=8))])
```

[5]:
```
linear_model = LinearRegression()
linear_model.fit(X_train, y_train)
```

[5]:
```
LinearRegression()
```

1. Work with your partner to create an ensemble model from your two models using **voting**.

Use cross-validation to see if the ensemble model is better than your individual models.

[6]:
```
# YOUR CODE HERE. ADD CELLS AS NEEDED

ensemble_model = VotingRegressor([
    ("linear", linear_model),
    ("knn", knn_model)],
    weights = [0.2, 0.8]
)
ensemble_model.fit(X_train, y_train)
ensemble_model.predict(X_test)
```

[6]:
```
array([198412.58865162, 130754.73068638, 155956.20086762, …,
       201072.44316882, 246105.24609227, 170764.2961691 ])
```

[7]:
```
for model in [linear_model, knn_model, ensemble_model]:
    print(-cross_val_score(model, X=X_train, y=y_train, cv=10,
                           scoring="neg_root_mean_squared_error").mean())
```

```
45365.84605381214
39772.88073764751
39510.32538124725
```

2. Work with your partner to create an ensemble model from your two models using **stacking**.

Use cross-validation to see if the ensemble model is better than your individual models.

[8]:
```
stacking_model = StackingRegressor([
    ("linear", linear_model),
    ("knn", knn_model)],
    final_estimator=LinearRegression()
)
stacking_model.fit(X=X_train, y=y_train)
```

```
stacker = stacking_model.final_estimator_
stacker.intercept_, stacker.coef_
```

[8]:
```
(-8436.103699506813, array([0.1954818 , 0.85714254]))
```

[9]:
```
-cross_val_score(stacking_model, X=X_train, y=y_train, cv=10,
                 scoring="neg_root_mean_squared_error").mean()
```

[9]:
```
39488.83080967234
```

3. In Discord, describe your model that resulted in your lowest MSE: features, method (linear or kNN), value of k if applicable, ensemble method etc. What was the test RMSE?

YOUR RESPONSE HERE.

Ensemble method resulted in a lower RMSE than the individual models. The ensemble method used was stacking. The base models were linear regression and kNN. The final estimator model was linear regression. The features used were the same as the ones used in the individual models. The test RMSE was 39488.

1.2 Restaurant Tips Data

The file `tips.csv` is an older data set containing information about dining parties at a restaurant, including the amount of the tips paid to the waiter. Our goal is to create a model for predicting the amount of the tip.

[10]:
```
df_tips = pd.read_csv("https://raw.githubusercontent.com/kevindavisross/data301/
 ↪main/data/tips.csv")

df_tips
```

[10]:
```
     total_bill   tip     sex smoker   day    time  size
0         16.99  1.01  Female     No   Sun  Dinner     2
1         10.34  1.66    Male     No   Sun  Dinner     3
2         21.01  3.50    Male     No   Sun  Dinner     3
3         23.68  3.31    Male     No   Sun  Dinner     2
4         24.59  3.61  Female     No   Sun  Dinner     4
..          …     …       …      …     …       …
239       29.03  5.92    Male     No   Sat  Dinner     3
240       27.18  2.00  Female    Yes   Sat  Dinner     2
241       22.67  2.00    Male    Yes   Sat  Dinner     2
242       17.82  1.75    Male     No   Sat  Dinner     2
243       18.78  3.00  Female     No  Thur  Dinner     2

[244 rows x 7 columns]
```

1. Use the data to determine a model for predicting the amount of the tip. In Discord, describe your model: features, method (linear or kNN), value of k if applicable, ensemble method etc. What was the test RMSE?

Note: this will be a good review exercise; can you implement all the relevant steps?

[11]:
```python
# YOUR CODE HERE. ADD CELLS AS NEEDED
df_tips_train = df_tips.loc[:122].copy()
df_tips_test = df_tips.loc[122:].copy()
X_train = df_tips_train[['total_bill', 'sex', 'smoker', 'day', 'time', 'size']]
y_train = df_tips_train["tip"]

X_test = df_tips_test[['total_bill', 'sex', 'smoker', 'day', 'time', 'size']]
```

[12]:
```python
from sklearn.compose import make_column_transformer
from sklearn.preprocessing import OneHotEncoder

# define the column transformer
preprocessor = make_column_transformer(
    (StandardScaler(), ['total_bill', 'size']),
    (OneHotEncoder(handle_unknown="ignore"),
     ['sex', 'smoker', 'day', 'time'])
)

# define the KNN model
knn_model = make_pipeline(
    preprocessor,
    KNeighborsRegressor(n_neighbors=8, metric='manhattan')
)

# define the linear regression model
linear_model = make_pipeline(
    preprocessor,
    LinearRegression()
)

# define the ensemble model
stacking_model2 = StackingRegressor([
    ('knn', knn_model),
    ('linear', linear_model)],
    final_estimator=LinearRegression()
)

# fit the ensemble model
stacking_model2.fit(X_train, y_train)
```

[12]:
```
StackingRegressor(estimators=[('knn',
                               Pipeline(steps=[('columntransformer',
ColumnTransformer(transformers=[('standardscaler',
 StandardScaler(),
 ['total_bill',
```

```
   'size']),
 ('onehotencoder',
  OneHotEncoder(handle_unknown='ignore'),
  ['sex',
   'smoker',
   'day',
   'time'])])),
                                                ('kneighborsregressor',
 KNeighborsRegressor(metric='manhattan',
 n_neighbors=8))])),
                               ('linear',
                                Pipeline(steps=[('columntransformer',
 ColumnTransformer(transformers=[('standardscaler',
  StandardScaler(),
  ['total_bill',
   'size']),
 ('onehotencoder',
  OneHotEncoder(handle_unknown='ignore'),
  ['sex',
   'smoker',
   'day',
   'time'])])),
                                                ('linearregression',
                                                 LinearRegression())]))],
                  final_estimator=LinearRegression())
```

[13]:
```
stacking_model2.predict(X_test)
```

[13]:
```
array([2.58034784, 2.7507063 , 2.32116556, 4.23241935, 2.00173389,
       2.52442083, 2.21028135, 3.47885717, 3.05629035, 3.09452655,
       2.18911255, 2.29898872, 2.89659243, 1.92097438, 2.10443734,
       2.48950795, 2.71509574, 2.38971216, 2.81695741, 4.7660943 ,
       5.40586214, 3.96888163, 2.71212143, 1.90484577, 2.97448613,
       2.25967523, 2.12874671, 1.89992203, 2.58674429, 2.49780783,
       2.95082818, 3.70635316, 3.22451088, 4.19097769, 6.1757697 ,
       3.66225026, 2.42184198, 2.89888405, 3.39890149, 2.45043004,
       2.75024731, 2.56368551, 2.79374655, 3.66798483, 3.24523956,
       4.43128062, 1.95703255, 1.9610647 , 6.12085764, 2.58341259,
       1.87595949, 4.32348765, 2.8306983 , 4.42933166, 2.93271737,
       2.59663309, 2.00378255, 4.60153463, 4.68701498, 3.46404841,
       5.72812779, 3.53190911, 5.19829327, 3.36609239, 3.18221336,
       4.31348895, 2.90887498, 3.4947371 , 2.71851822, 3.00989057,
       3.95745702, 2.66267775, 2.77356196, 1.90496222, 2.14824322,
       5.4352366 , 2.33299842, 2.46779134, 3.03383462, 2.31048655,
       2.33299842, 2.66654331, 3.26058294, 2.72005703, 3.69962476,
       4.95264573, 3.43115948, 2.1735892 , 4.04643883, 3.68201685,
       5.95259075, 2.22270212, 3.77181472, 2.18722045, 3.94996864,
```

```
       2.15802046, 1.76992574, 4.0156525 , 2.26014895, 2.2958814 ,
       1.8549577 , 2.63549077, 2.38674611, 2.57989082, 1.96567378,
       3.17906387, 2.36489502, 3.10429338, 3.49250566, 2.62081442,
       2.19655264, 2.11187743, 2.55518752, 2.04131475, 2.25983232,
       4.29132753, 4.58166943, 3.98986834, 3.61473263, 3.26987336,
       2.80814967, 2.89750782])
```

[14]:
```
-cross_val_score(stacking_model2, X=X_train, y=y_train, cv=5,
                 scoring="neg_root_mean_squared_error").mean()
```

[14]:
```
0.9065902863098781
```

2. What is a rough benchmark for RMSE in this context? Is your model's RMSE a substantial improvement over this benchmark?

[15]:
```
df_tips["tip"].std()
```

[15]:
```
1.3836381890011822
```

A rough benchmark for the RMSE in this context would be the standard deviation of the tips, since it gives the variability of the tips. Our model's RMSE is lower which means that its predictions are on averages closer to the actual values that if we were to use the mean of the tips.