Linear Model

Sreejith Sreekumar

August 13, 2021

Contents

- Sklearn's *'train_test_split'* should not be used as data is time series
- Pythons' indexing is used instead
- In stats models, a constant has to be added manually to include the bias term
- In linear regression p-values are obtained as a result of a t-test on the co-efficients
- p-value is the percent chance that a coefficient is actually zero. i,e it has no effect on the target

1 Splitting the data

```
# Import the statsmodels.api library with the alias sm
import statsmodels.api as sm

# Add a constant to the features
linear_features = sm.add_constant(features)

# Create a size for the training set that is 85% of the total number of samples
train_size = int(0.85 * features.shape[0])
train_features = linear_features[:train_size]
```

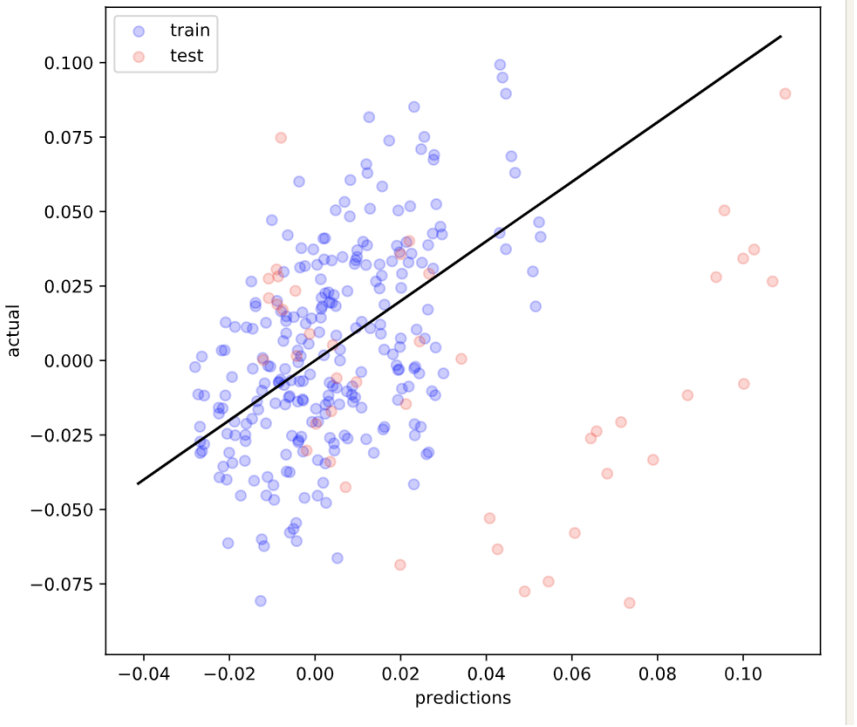
```
train_targets = targets[:train_size]
test_features = linear_features[train_size:]
test_targets = targets[train_size:]
print(linear_features.shape, train_features.shape, test_features.shape)
```

2 Building the model

```
# Create the linear model and complete the least squares fit
model = sm.OLS(train_targets, train_features)
results = model.fit()  # fit the model
print(results.summary())

# examine pvalues
# Features with p <= 0.05 are typically considered significantly different from 0
print(results.pvalues)

# Make predictions from our model for train and test sets
train_predictions = results.predict(train_features)
test_predictions = results.predict(test_features)
```


train
test
0.100
0.075
0.050
0.025
0.000
−0.025
−0.050
−0.075
actual
−0.04
−0.02
0.00
0.02
0.04
0.06
0.08
0.10
predictions