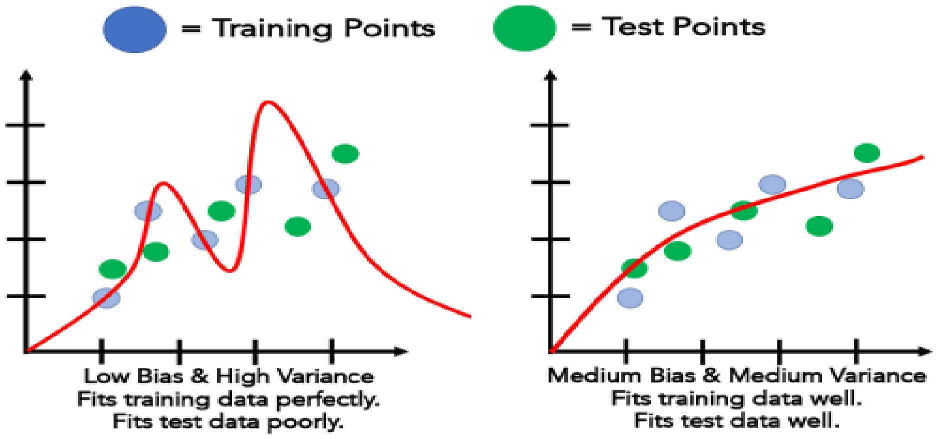
Regularized Regression (Ridge & Lasso)

Regularization: a technique used to avoid overfitting



1. Ridge Regression

Loss Function = OLS **(Ordinary least squares)** loss Function + **Penalty**

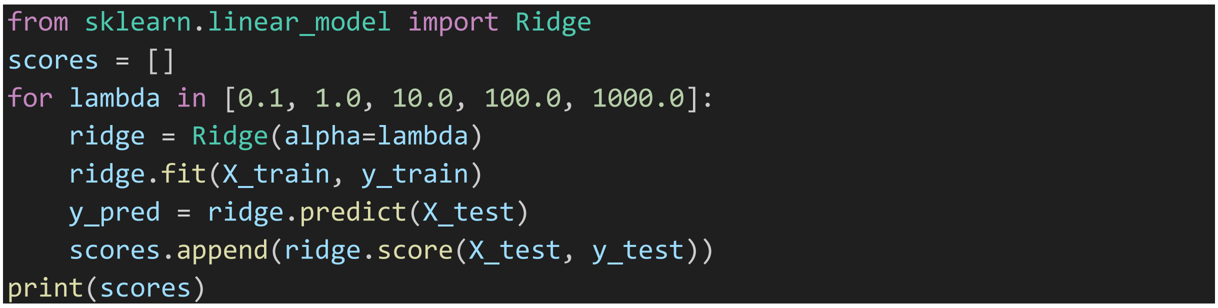
Ridge

$$\text{Penalty} = \lambda * (slope)^2$$

Ridge Penalizes Large positive or negative coefficients

λ (hyperparameter), Controls model complexity:

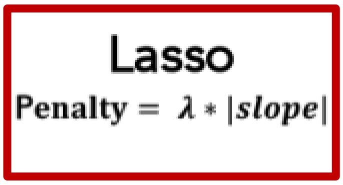
- If λ = 0, ∴ OLS => (Leads to Overfitting)
- If λ is very High, ∴ Leads to Underfitting



```
from sklearn.linear_model import Ridge
scores = []
for lambda in [0.1, 1.0, 10.0, 100.0, 1000.0]:
    ridge = Ridge(alpha=lambda)
    ridge.fit(X_train, y_train)
    y_pred = ridge.predict(X_test)
    scores.append(ridge.score(X_test, y_test))
print(scores)
```

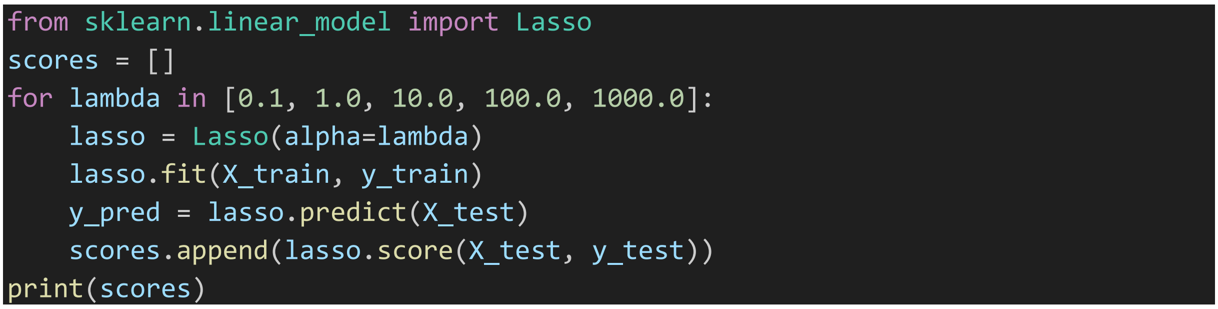
2. Lasso Regression

Loss Function = OLS **(Ordinary least squares)** loss Function + **Penalty**



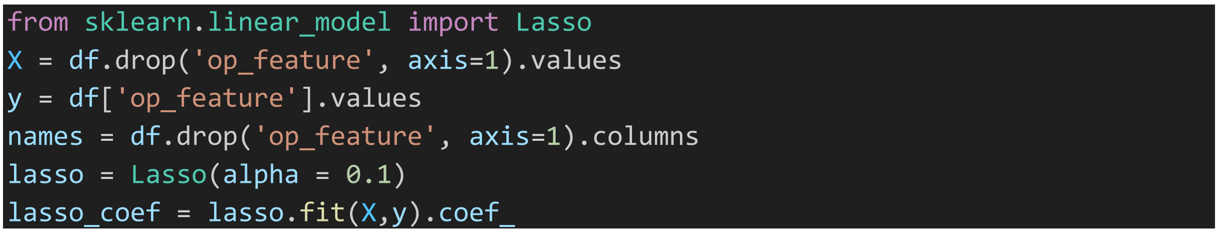
Lasso

Penalty = $\lambda * |slope|$



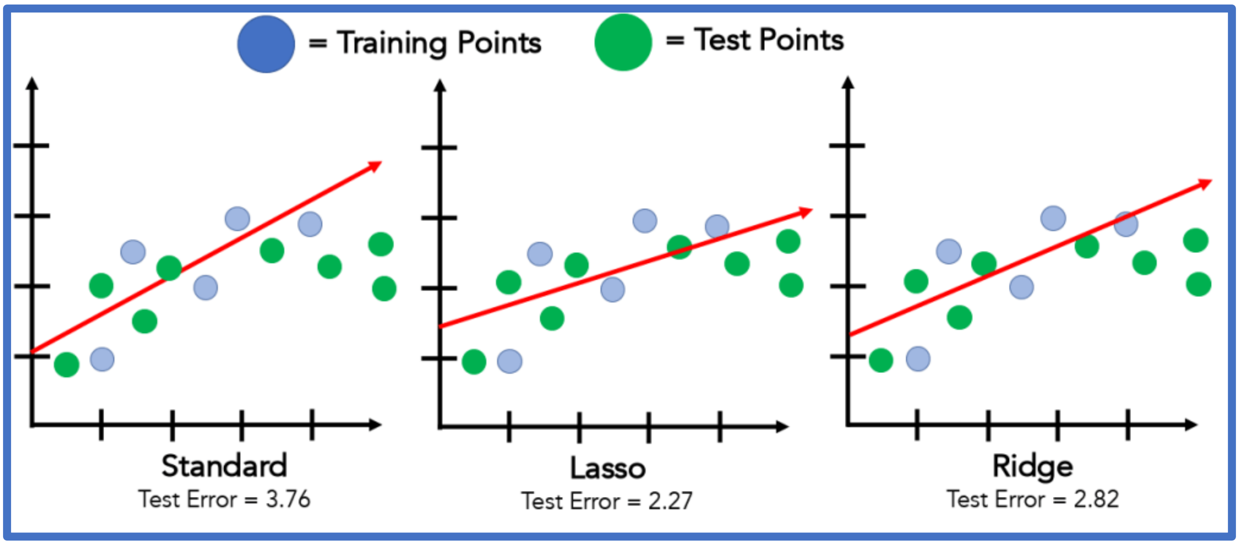
```
from sklearn.linear_model import Lasso
scores = []
for lambda in [0.1, 1.0, 10.0, 100.0, 1000.0]:
    lasso = Lasso(alpha=lambda)
    lasso.fit(X_train, y_train)
    y_pred = lasso.predict(X_test)
    scores.append(lasso.score(X_test, y_test))
print(scores)
```

Lasso is used for Feature Selection, ∴ it shrinks the coefficients of less important features to 0.



```
from sklearn.linear_model import Lasso
X = df.drop('op_feature', axis=1).values
y = df['op_feature'].values
names = df.drop('op_feature', axis=1).columns
lasso = Lasso(alpha = 0.1)
lasso_coef = lasso.fit(X,y).coef_
```

Example:



● = Training Points
● = Test Points
Standard
Test Error = 3.76
Lasso
Test Error = 2.27
Ridge
Test Error = 2.82