Home › Tutorials › Data Science

Lasso and Ridge Regression Tutorial

Learn about the lasso and ridge techniques of regression. Compare and analyse the methods in detail.

Mar 2022 · 10 min read

DataCamp Team

Making data science accessible to everyone

TOPICS

Data Science

Machine Learning

Introducing Linear Models

Linear regression is a type of linear model that is considered the most basic and commonly used predictive algorithm. This can not be dissociated from its simple, yet effective architecture. A linear model assumes a linear relationship between input variable(s) x and an output variable y. The equation for a linear model looks like this:

$$\hat{y} = w[0] \times x[0] + w[1] \times x[1] + ... + w[n] \times x[n] + b \qquad \text{Eqn 1.1}$$

In this equation 1.1, we show a linear model with n number of features. w is considered the coefficient (or weights) assigned to each feature - an indicator of their significance to the outcome y. For example, we assume that temperature is a larger driver of ice cream sales than whether it's a public holiday. The weight assigned to temperature in our linear model will be larger than the public holiday variable.

The goal for a linear model then becomes to optimize the weight (b) via the cost function in equation 1.2. The cost function calculates the error between predictions and actual values, represented as a single real-valued number. The cost function is the average error across n samples in the dataset, represented below as:

Cost function of a linear regression model

In the equation above, yi is the actual value and that is the predicted value from our linear equation, where M is the number of rows and P is the number of features.

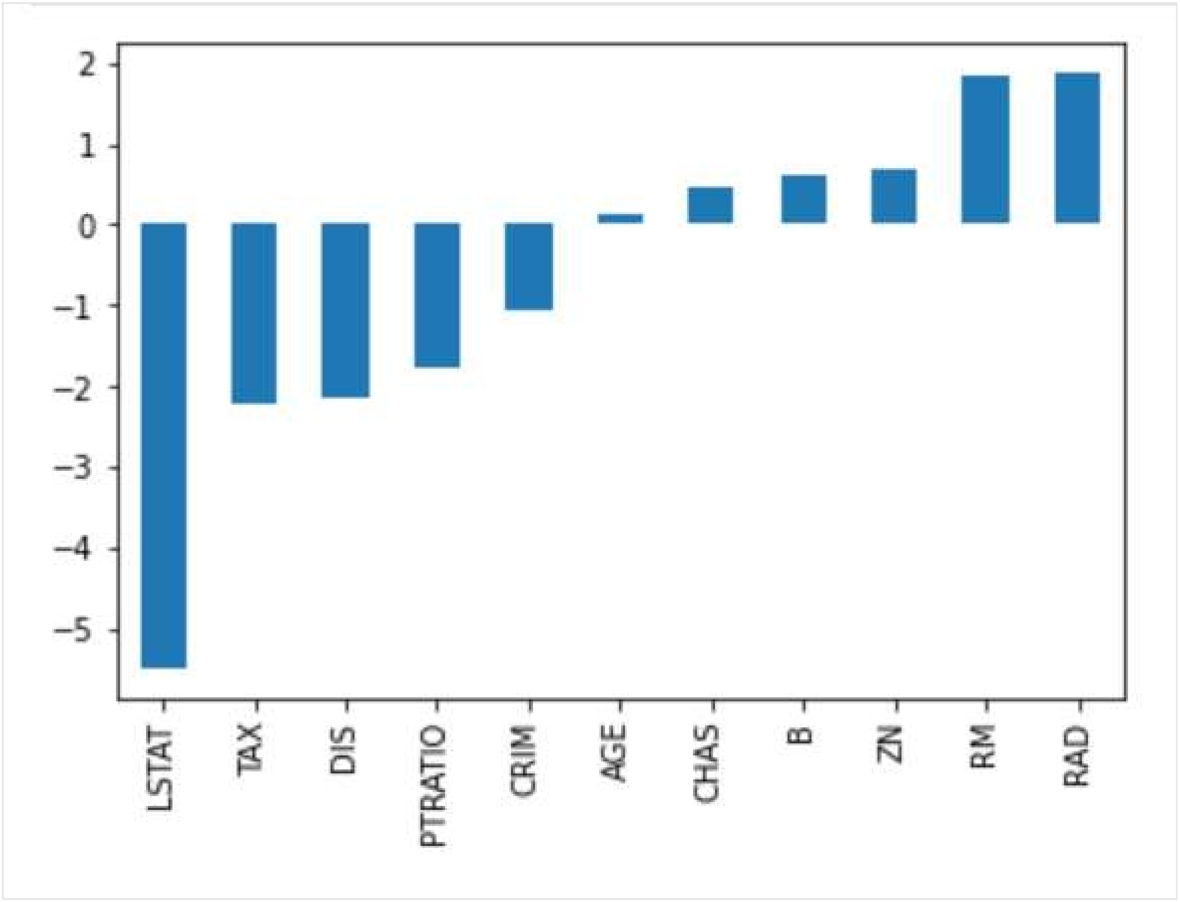
Regularization

When it comes to training models, there are two major problems one can encounter: **overfitting** and underfitting.

- Overfitting happens when the model performs well on the training set but not so well on unseen (test) data.
- Underfitting happens when it neither performs well on the train set nor on the test set.

Particularly, regularization is implemented to avoid overfitting of the data, especially when there is a large variance between train and test set performances. With regularization, the number of features used in training is kept constant, yet the magnitude of the coefficients (w) as seen in equation 1.1, is reduced.

Consider the image of coefficients below to predict house prices. While there are quite a number of predictors, RM and RAD have the largest coefficients. The implication of this will be that housing prices will be driven more significantly by these two features leading to overfitting, where generalizable patterns have not been learned.



There are different ways of reducing model complexity and preventing overfitting in linear models. This includes ridge and lasso regression models.

Introduction to Lasso Regression

This is a regularization technique used in feature selection using a Shrinkage method also referred to as the **penalized regression method**. Lasso is short for **L**east **A**bsolute **S**hrinkage and **S**election **O**perator, which is used both for regularization and model selection. If a model uses the **L1 regularization** technique, then it is called lasso regression.

Lasso Regression for Regularization

In this shrinkage technique, the coefficients determined in the linear model from equation 1.1. above are shrunk towards the central point as the mean by introducing a penalization factor called the alpha α (or sometimes lamda) values.

$$L_{lasso}(\hat{\beta}) = \sum_{i=1}^{n}(y_i - x_i'\hat{\beta})^2 + \lambda \sum_{j=1}^{m}|\hat{\beta}_j|.$$

Alpha (α) is the penalty term that denotes the amount of shrinkage (or constraint) that will be implemented in the equation. With alpha set to zero, you will find that this is the equivalent of the linear regression model from equation 1.2, and a larger value penalizes the optimization function. Therefore, lasso regression shrinks the coefficients and helps to reduce the model complexity and multi-collinearity.

Alpha (α) can be any real-valued number between zero and infinity; the larger the value, the more aggressive the penalization is.

Lasso Regression for Model Selection

Due to the fact that coefficients will be shrunk towards a mean of zero, less important features in a dataset are eliminated when penalized. The shrinkage of these coefficients based on the alpha value provided leads to some form of automatic feature selection, as input variables are removed in an effective approach.

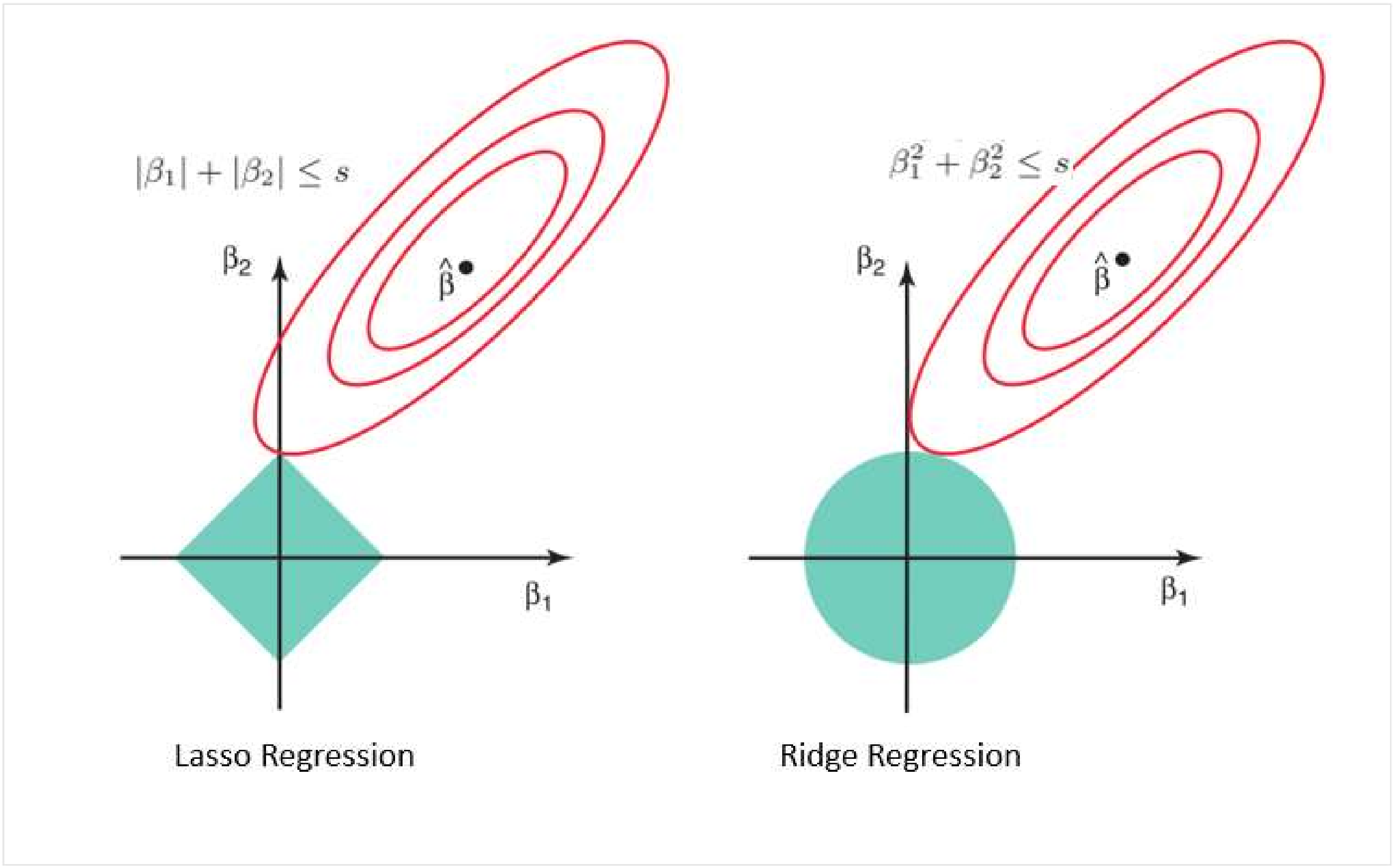
Ridge Regression

Similar to the lasso regression, ridge regression puts a similar constraint on the coefficients by introducing a penalty factor. However, while lasso regression takes the magnitude of the coefficients, ridge regression takes the square.

$$L_{hridge}(\hat{\beta}) = \sum_{i=1}^{n}(y_i - x_i'\hat{\beta})^2 + \lambda \sum_{j=1}^{m} w_j \hat{\beta}_j^2.$$

Ridge regression is also referred to as **L2 Regularization**.

Why Lasso can be Used for Model Selection, but not Ridge Regression



Source

Considering the geometry of both the lasso (left) and ridge (right) models, the elliptical contours (red circles) are the cost functions for each. Relaxing the constraints introduced by the penalty factor leads to an increase in the constrained region (diamond, circle). Doing this continually, we will hit the center of the ellipse, where the results of both lasso and ridge models are similar to a linear regression model.

However, both methods determine coefficients by finding the first point where the elliptical contours hit the region of constraints. Since lasso regression takes a diamond shape in the plot for the constrained region, each time the elliptical regions intersect with these corners, at least one of the coefficients becomes zero. This is impossible in the ridge regression model as it forms a circular shape and therefore values can be shrunk close to zero, but never equal to zero.

never equal to zero.

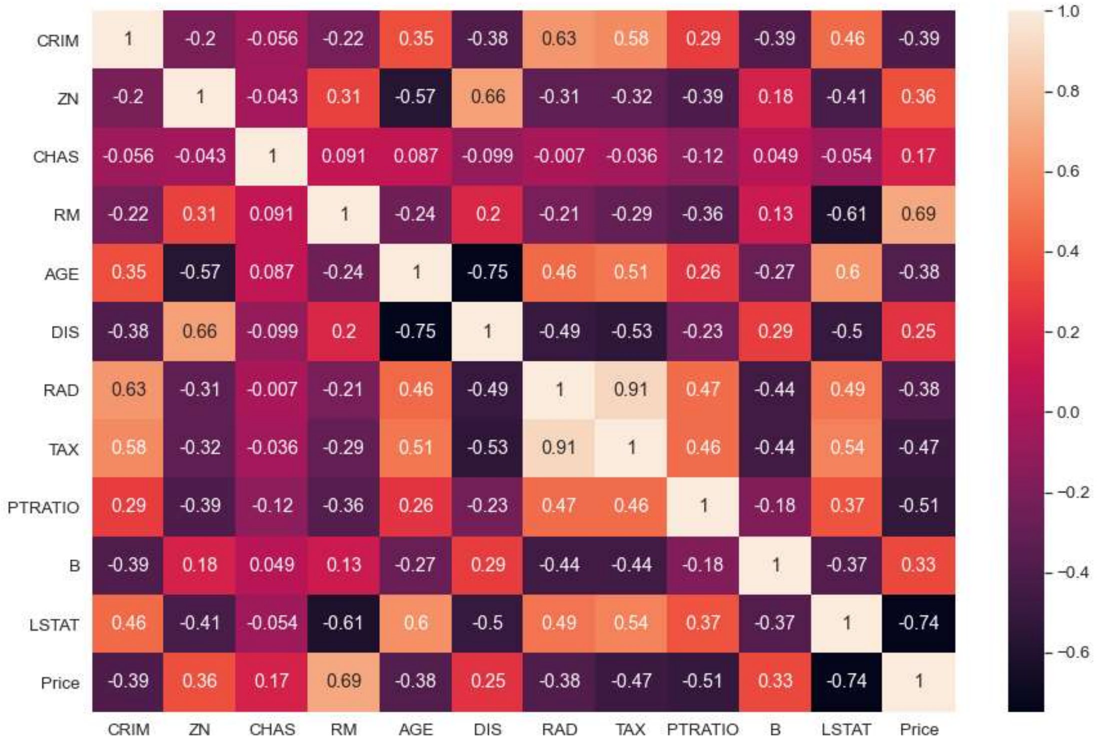
Python Implementation

For this implementation, we will use the Boston housing dataset found in Sklearn. What we intend to see is:

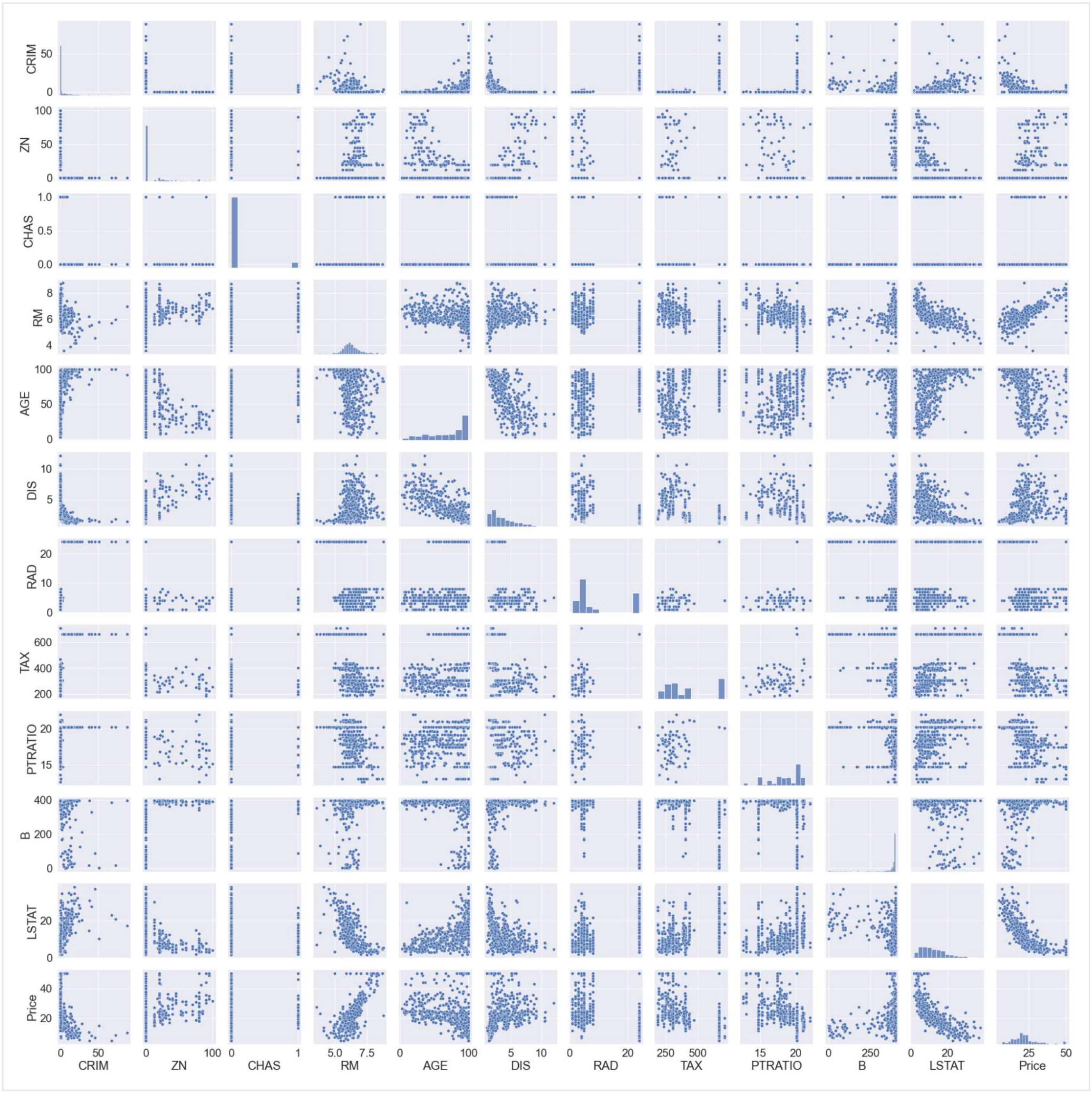
1. How to perform ridge and lasso regression in **Python**
2. Compare the results with a linear regression model

Data Importation and EDA

```
#libraries
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.linear_model import Ridge, RidgeCV, Lasso
from sklearn.preprocessing import StandardScaler
from sklearn.datasets import load_boston

#data
boston = load_boston()
boston_df=pd.DataFrame(boston.data,columns=boston.feature_names)
#target variable
boston_df['Price']=boston.target
#preview
boston_df.head()

#Exploration
plt.figure(figsize = (10, 10))
sns.heatmap(boston_df.corr(), annot = True)
```



```
#There are cases of multicolinearity, we will drop a few columns
boston_df.drop(columns = ["INDUS", "NOX"], inplace = True)

#pairplot
sns.pairplot(boston_df)

#we will log the LSTAT Column
boston_df.LSTAT = np.log(boston_df.LSTAT)
```



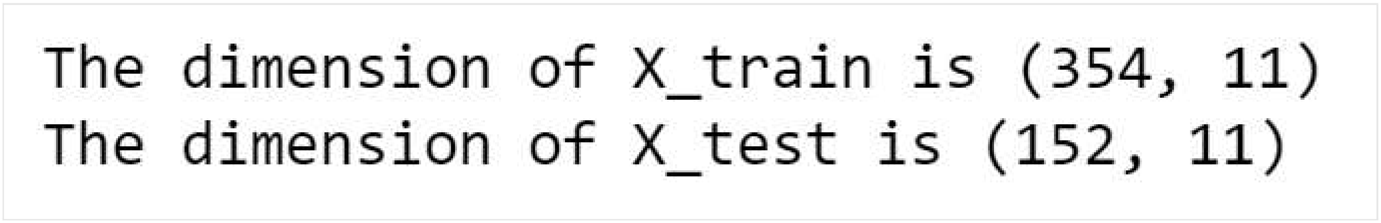
Note that we logged the LSTAT column as it doesn't have a linear relationship with the price column. Linear models assume a linear relationship between x and y variables.

Data Splitting and Scaling

```
#preview
features = boston_df.columns[0:11]
target = boston_df.columns[-1]

#X and y values
```

```
X = boston_df[features].values
y = boston_df[target].values

#splot
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, rando

print("The dimension of X_train is {}".format(X_train.shape))
print("The dimension of X_test is {}".format(X_test.shape))
#Scale features
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

Output:

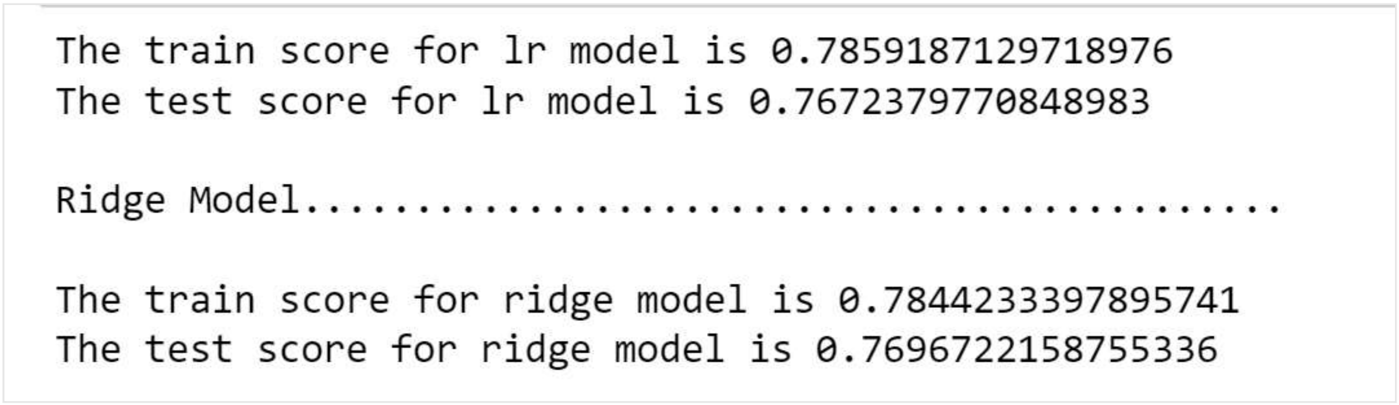


```
The dimension of X_train is (354, 11)
The dimension of X_test is (152, 11)
```

Linear and Ridge Regression Models

We will build a linear and a ridge regression model and then compare the coefficients in a plot. The score of the train and test sets will also help us evaluate how well the model performs.

```
#Model
lr = LinearRegression()

#Fit model
lr.fit(X_train, y_train)

#predict
#prediction = lr.predict(X_test)

#actual
actual = y_test

train_score_lr = lr.score(X_train, y_train)
test_score_lr = lr.score(X_test, y_test)

print("The train score for lr model is {}".format(train_score_lr))
print("The test score for lr model is {}".format(test_score_lr))


#Ridge Regression Model
ridgeReg = Ridge(alpha=10)

ridgeReg.fit(X_train,y_train)

#train and test scorefor ridge regression
train_score_ridge = ridgeReg.score(X_train, y_train)
test_score_ridge = ridgeReg.score(X_test, y_test)

print("\nRidge Model............................................\n")
print("The train score for ridge model is {}".format(train_score_ridge))
print("The test score for ridge model is {}".format(test_score_ridge))
```

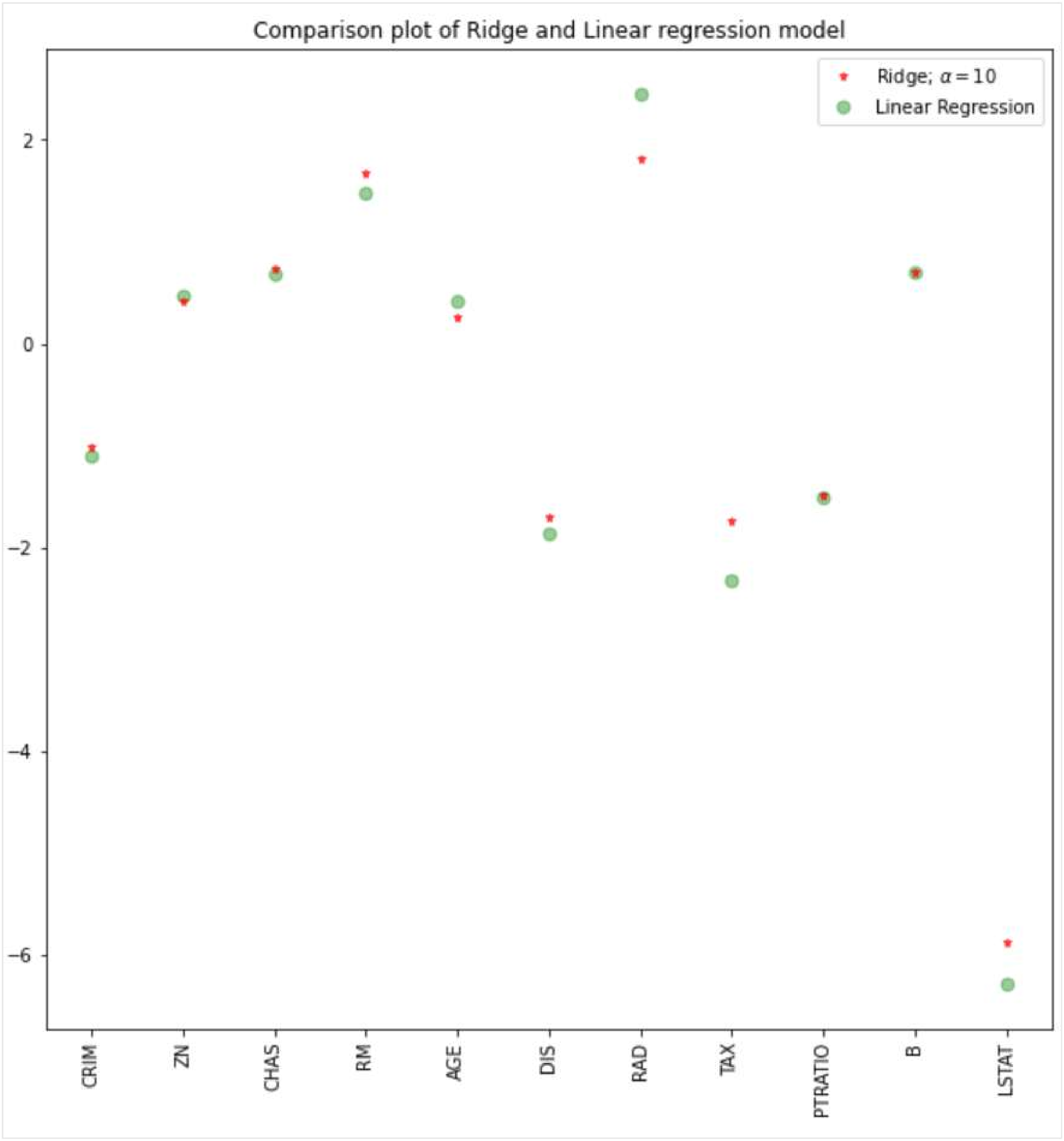


```
The train score for lr model is 0.7859187129718976
The test score for lr model is 0.7672379770848983

Ridge Model............................................

The train score for ridge model is 0.7844233397895741
The test score for ridge model is 0.7696722158755336
```

Using an alpha value of 10, the evaluation of the model, the train, and test data indicate better performance on the ridge model than on the linear regression model.

We can also plot the coefficients for both the linear and ridge models.

```
plt.figure(figsize = (10, 10))
plt.plot(features,ridgeReg.coef_,alpha=0.7,linestyle='none',marker='*',markers
#plt.plot(rr100.coef_,alpha=0.5,linestyle='none',marker='d',markersize=6,color
plt.plot(features,lr.coef_,alpha=0.4,linestyle='none',marker='o',markersize=7,
plt.xticks(rotation = 90)
plt.legend()
plt.show()
```



Lasso Regression

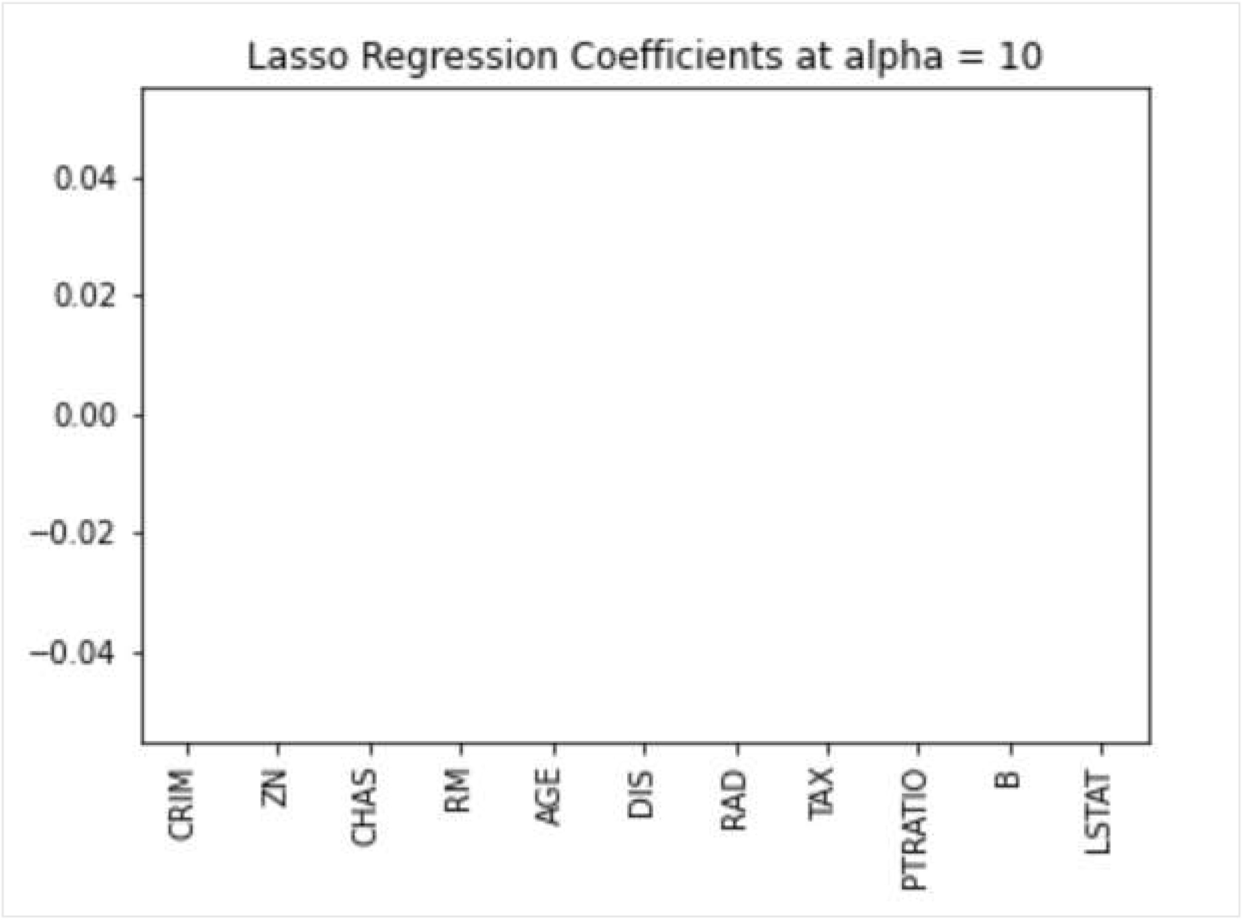
```
#Lasso regression model
print("\nLasso Model............................................\n")
lasso = Lasso(alpha = 10)
lasso.fit(X_train,y_train)
train_score_ls =lasso.score(X_train,y_train)
```

```
test_score_ls =lasso.score(X_test,y_test)

print("The train score for ls model is {}".format(train_score_ls))
print("The test score for ls model is {}".format(test_score_ls))
```

```
Lasso Model.........................................

The train score for ls model is 0.0
The test score for ls model is -0.0030704836212473996
```

We can visualize the coefficients too.

```
pd.Series(lasso.coef_, features).sort_values(ascending = True).plot(kind = "ba
```



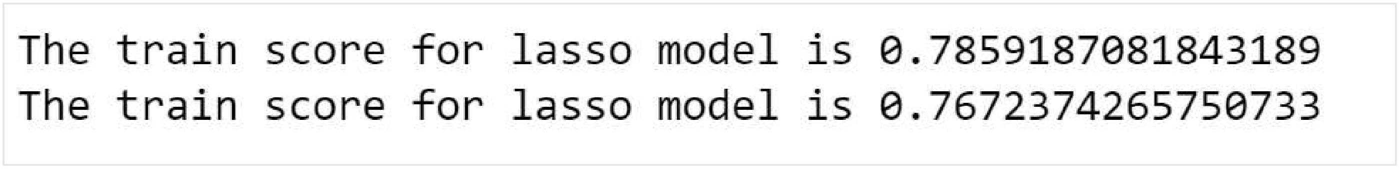
Earlier on, we established that the lasso model can inert to zero due to the diamond shape of the constraint region. In this case, using an alpha value of 10 over penalizes the model and shrinks all the values to zero. We can see this effectively by visualizing the coefficients of the model as shown in the figure above.

Selecting Optimal Alpha Values Using Cross-Validation in Sklearn

We may need to try out different alpha values to find the optimal constraint value. For this case, we can use the cross-validation model in the **sklearn package**. This will try out different combinations of alpha values and then choose the best model.

```
#Using the linear CV model
from sklearn.linear_model import LassoCV

#Lasso Cross validation
lasso_cv = LassoCV(alphas = [0.0001, 0.001,0.01, 0.1, 1, 10], random_state=0).


#score
print(lasso_cv.score(X_train, y_train))
print(lasso_cv.score(X_test, y_test))
```

Output:

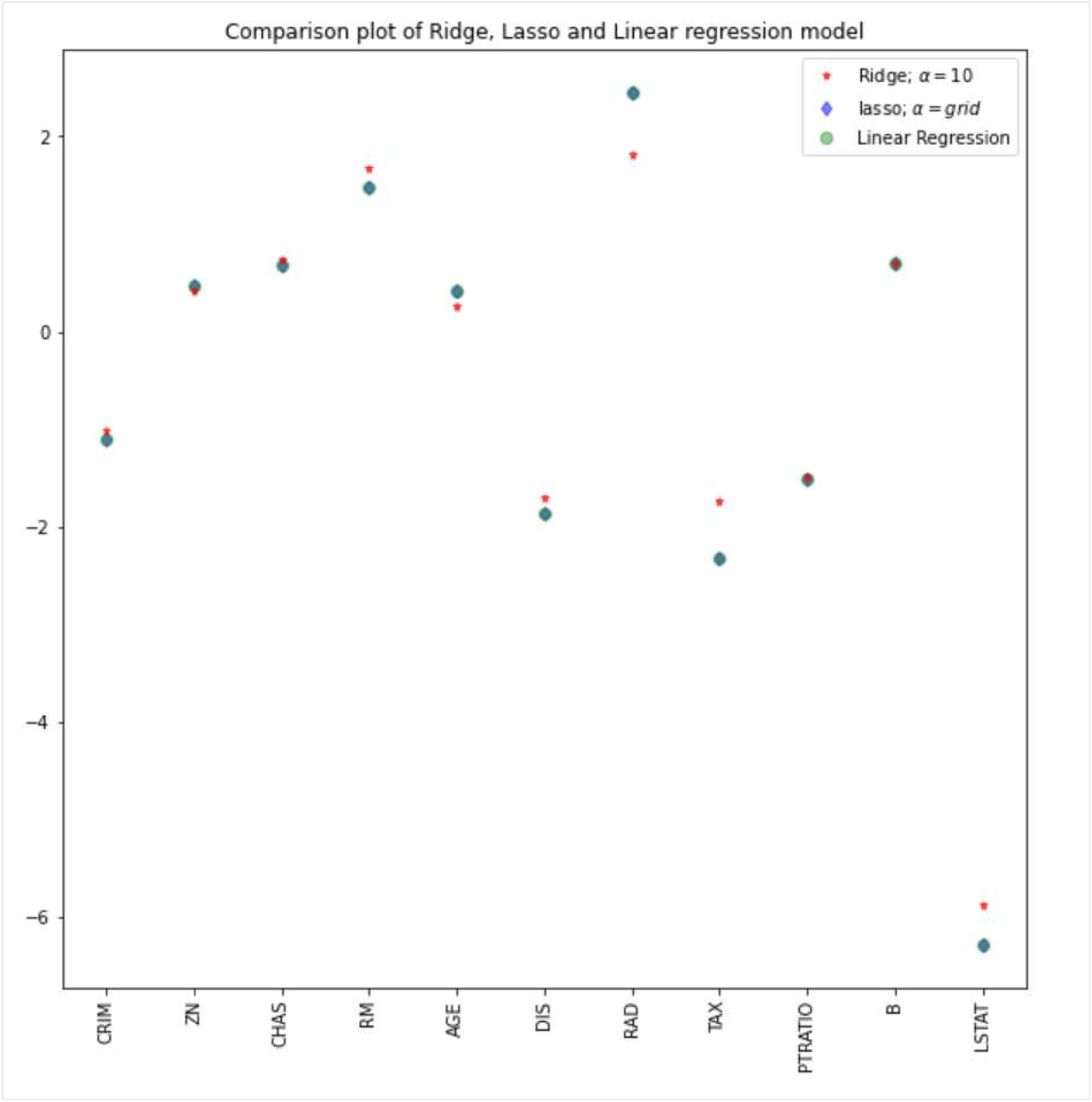


```
The train score for lasso model is 0.7859187081843189
The train score for lasso model is 0.7672374265750733
```

The model will be trained on different alpha values that I have specified in the LassoCV function. We can observe a better performance of the model, removing the tedious effort of manually changing alpha values.

We can compare the coefficients from the lasso model with the rest of the models (linear and ridge).

```
#plot size
plt.figure(figsize = (10, 10))
#add plot for ridge regression
plt.plot(features,ridgeReg.coef_,alpha=0.7,linestyle='none',marker='*',markers

#addd plot for lasso regression
plt.plot(lasso_cv.coef_,alpha=0.5,linestyle='none',marker='d',markersize=6,col
#add plot for linear model
plt.plot(features,lr.coef_,alpha=0.4,linestyle='none',marker='o',markersize=7,

#rotate axis
plt.xticks(rotation = 90)
plt.legend()
plt.title("Comparison plot of Ridge, Lasso and Linear regression model")
plt.show()
```



Note: A similar approach could be employed for the ridge regression model, which could lead to better results. In the sklearn package, the function RidgeCV performs similarly.

```
#Using the linear CV model
from sklearn.linear_model import RidgeCV

#Lasso Cross validation
ridge_cv = RidgeCV(alphas = [0.0001, 0.001,0.01, 0.1, 1, 10]).fit(X_train, y_t

#score
print("The train score for ridge model is {}".format(ridge_cv.score(X_train, y
print("The train score for ridge model is {}".format(ridge_cv.score(X_test, y_
```

Click here to learn about other kinds of regression you can implement in Python.

Conclusion

We have seen an implementation of ridge and lasso regression models and the theoretical and mathematical concepts behind these techniques. Some of the key takeaways from this tutorial include:

1. The cost function for both ridge and lasso regression are similar. However, ridge regression takes the square of the coefficients and lasso takes the magnitude.
2. Lasso regression can be used for automatic feature selection, as the geometry of its constrained region allows coefficient values to inert to zero.
3. An alpha value of zero in either ridge or lasso model will have results similar to the regression model.
4. The larger the alpha value, the more aggressive the penalization.

You can find a more robust and complete notebook for the python implementation **here**, or do a deep dive into regressions with our **Introduction to Regression in Python** course.

TOPICS

Data Science Machine Learning

For Classrooms

Discounts, Promos & Sales

DataCamp Donates

SUPPORT

Help Center

Become an Instructor

Become an Affiliate

ABOUT

About Us

Learner Stories

Careers

Press

Leadership

Contact Us

Download on the App Store

GET IT ON Google Play

Privacy Policy Cookie Notice Do Not Sell My Personal Information Accessibility Security

Terms of Use