Predicting car prices using various regression models and comparing model performances.

Nikita Kohli (T00687733), Sarah Chopra(T00684826), and Robin Teotia (T00671961)

Dr. Mateen Shaikh

Master of Science, Data Science

Faculty of Mathematics & Statistics

Linear Models

Thompson Rivers University

April 14. 2022

Abstract

Over 70 million passenger automobiles were produced in 2016, indicating that car production has been gradually expanding over the last decade. This study employed recursive feature elimination (RFE) to get the significant features that were used to perform multiple linear regression (MLR), principal component regression (PCR), and partial least squares regression (PLSR) methods to predict the car prices. The goal was to conduct a comparative study on performance of regression, based on both unsupervised and supervised models, which was done by using R-squared values. Each model was trained on the car dataset taken from the UCI repository. As a result, PLSR and MLR comparatively performed well with R-squared as 75.9% followed by PCR with R-squared as 74.6%. On the other hand, before feature selection using RFE, the MLR model was observed to overfit the data.

Keywords: Prediction; Multiple Linear Regression; Recursive Feature Elimination; Decision Trees; Principal Component Regression; Partial Least Squares Regression

Introduction

The automobile market is a growing business with a market value that has nearly doubled in the previous few years. This study used an accessible dataset to predict the price of cars using the available variables. This process will help understanding how exactly prices vary with the features. Therefore, the required information can be used in the car's design, production process, and other factors to achieve specified price targets. The main goal of this research is to evaluate and compare the accuracy of three different applied models for predicting the price of a car.

The data set was gathered from UCI repository (Schlimmer, 1987), which has 201 records of cars with prices determined and 29 features. The data collection consists of almost equal numbers of categorical features and quantitative attributes.

For this project, the price column was used as the target variable. A high correlation was observed between the features. For categorical variables, a dummy coding was applied which increased the features to 79 as the level of each category was added to the data set. Since, linear regression cannot fit the NaN values, hence, the four rows containing those values were manually removed. The final data set with 79 columns and 197 instances was then split into dependent (**Y**) and independent variables(**X**). Using these attributes, the price was then predicted using Python.

Methodology

When the statistical models are developed, the fitting of the data must be considered. The models may be overly influenced by the training data and so perform poorly on the test data set. This is referred to as overfitting and the same issue was observed in this project.

The hypothesis for this study was that PCR is a better model at predicting prices than MLR and PLSR. The data was split into training (80%) and testing (20%) samples through random sampling. In this section, the applied algorithms are explained.

Multiple Linear Regression

For automobiles dataset multiple linear regression was chosen because of the presence of a numerical target. Also, this study aims to know how the predictor variables like horsepower, engine-size etc. affect the price of a car. Post applying dummy coding to categorical variables, the MLR model was executed on the new set of features. The idea is to understand which variables can predict the price (James, et al., 2008). The model suggested features like normalized-losses, wheelbase, length, height, curb-weight, engine-size, bore, horsepower, peak-rpm, city-L/100km, make of the car are the significant features based upon the p-values. The equation of MLR:

$$y_i = \boldsymbol{\beta}_0 + \boldsymbol{\beta}_1 x_{i1} + \boldsymbol{\beta}_2 x_{i2} + \ldots \boldsymbol{\beta}_p x_{ip} + \varepsilon_i \text{ for } i = 1,2, \ldots n. \quad (1)$$

Here, n = number of observations (197), p is the number of features (78).

Recursive feature elimination

The weights are assigned to the features (for example, the coefficients of the linear model) by the provided external estimator. The purpose of recursive feature elimination (RFE) is to choose variables by recursively analyzing the smaller and smaller groups of features. First on the initial set of features the estimator is trained then each feature's importance is assessed using any specified property (such as coefficients or features importance) as mentioned by (Pedregosa et al., 2011). After that from the initial set of features, the features having the least importance are pruned. This technique is continued recursively on the pruned set until the required number of features is obtained. To be the estimator for the RFE, a supervised learning estimator is needed that provides information about the importance of the features. In this project as the task is regression, the Decision Trees for regression, support vector machines for regression, random forest method for regression etcetera can be used to select variable. In this study, the decision tree was used as an estimator for RFE.

Decision Tree

The algorithm is required to determine the splitting features and split points automatically, as well as the tree's shape. The methodology is mentioned by Hastie et al., (2008). Now let's assume R_1, R_2, R_3 ...R_M as M partition regions and in each region, the response was modelled as a constant c_m. The form $f(x) = \sum_{m=1}^{M} c_m \, I\,(x \in R_m)$ where the adopted criteria is to minimize the sum of squares $\sum(y_i - f(x_i))^2$ and where the c_m is just the average of y_i in the region R_m. This is given by the formula, $c_m = ave\,(y_i \,|x_i \in R_m)$. Here the algorithm acts as a greedy algorithm to

find the best binary partition in the sense of minimum sum of squares. Here, the algorithm starts with whole of the data, find a splitting variable j and splitting terminal s and forms the pairs of half-planes given as R_1 (j, s) = (X|X j ≤ s) and R_2 (j, s) = (X|X j > s). which are so formed to minimize the MSE given as, $min_{j,s} = [\min \sum_{x_i \in R_1}(y_i - c_{1i})^2 + \min \sum_{x_i \in R_2}(y_i - c_2)^2]$ where c_1 = ave (y_i |x_i ∈ R_1(j, s)) and c_2 = ave (y_i |x_i ∈ R_2(j, s)).

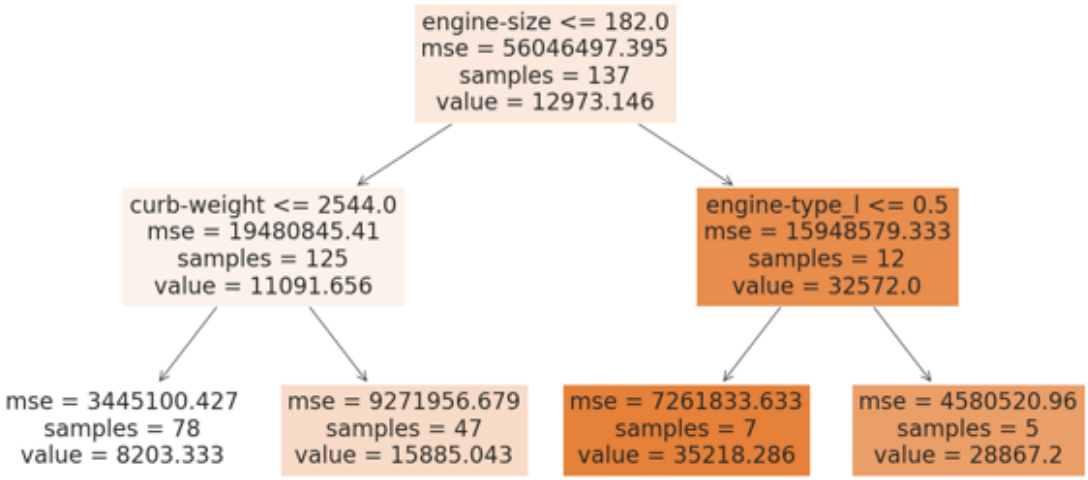

Figure 1. Decision Tree

The diagram of the tree in Figure 1 has three internal nodes and the count of terminal nodes is 4. Therefore, this tree gives rise to 4 regions, for example the region one can be represented as: R_1 = (X| engine-size ≤ 182.0, curb-weight ≤ 2544.0) as explained by James, et al., (2013) which states that the price of a car whose engine size is less than or equal to 182.0 units and curb-weight is less than or equal to 2544.0 units, have the price 8203.33 units[1]. The employed method of RFE with

[1] The UCI repository has not provided the information about the units for the target variable in the automobile data set.

estimator decision trees help us in selecting five features named as engine size, height, horsepower, curb weight, and engine-ohc.

Principal Component Regression

Principal Components Regression, PCR is an unsupervised dimension reduction approach that is based upon principal component analysis (James et al., 2013). For PCR, principal components are used as predictors in a regression model. The principal components are identified by linear combinations of features. A principal component outputs weight for each feature variable. These principal components explain the variation of features. These components are selected so that the cumulative variation explained by them is sufficient to explain feature variables. This approach also solves the collinearity issues that are found in MLR.

The algorithm of PCR used by Rong et al., (2021). was referred for the application. PCR is a regression method in two steps. The variables should be standardized. The first step is to perform a principal component analysis on predictor variables to reduce the dimensionality of the data and consider the structure of the design matrix. PCA extracts the most variation of a data set in an iterative process. The first component as the direction maximizing the spread of the observations is found which can be seen when the samples are projected on the component. The second component will also maximize the dispersion as well as will be orthogonal to the first component. The principal components obtained from PCA can be denoted as follows:

$PC_i = \sum_{i=0}^{k} \gamma_i X_i$ where X_i are the predictor variables and γ_i are loadings.

The second step of PCR is to do linear regression between the target on the selected principal components. The following equation shows that the result of PCR is a linear model using the components.

$Y_i = \sum_{i=0}^{j \leq k} \alpha_i (PC_i) + \varepsilon$ where α_i are the regression coefficient to be estimated.

Partial Least Squares Regression

Partial Least Squares (PLS) is a supervised approach hence it identifies the new features in a supervised way. The PLS approach attempts to find linear combinations that help explain both the response and the predictors. The algorithm of PLS regression used by Mehmood et al.,(2021) was referred for this study's application. For the PLSR algorithm it is assumed that $X \epsilon R^{n\times p}$ and $Y \epsilon R^{n\times 1}$ are input and output variables linked through a linear relationship $Y = \beta_0 + \beta_1 X + \varepsilon$. Initially, the variables can be centered and scaled into $X_0 = X - 1\bar{x}^T$ and $Y_0 = Y - 1\bar{y}^T$. For the number of components Q = 1,2, …, q, the algorithm follows as:

Step 1. The loading weights can be computed as $u_q = X_{q-1}^T Y_{q-1}$. The weights define the direction in the space spanned by X_{q-1} of maximum covariance with Y_{q-1} . Weights of loading are normalized by $\frac{u_q}{||u_q||} \rightarrow u_q$.

Step 2. The score vector is obtained by $s_q = X_{q-1} u_q$.

Step 3. The X-loadings and Y-loadings are computed by $I_q = X_{q-1}^T \frac{s_q}{s_q^T s_q}$ and $O_q = Y_{q-1}^T \frac{s_q}{s_q^T s_q}$.

Step 4. Updated variables can be obtained by $X_q = X_{q-1} - s_q I_q^T$ and $Y_q = Y_{q-1} - s_q O_q$.

Step 5. If Q ≤ p, then stop the iteration.

Matrices W, S, I can be used to store weights, scores, X- loadings respectively while vector O can be used to store Y-loadings obtained at each iteration. The PLSR-estimator for regression coefficients for linear model can be found by $\hat{\beta}_1 = W (I^T W)^{-1} I$ and $\hat{\beta}_0 = \bar{y} - \bar{x}\hat{\beta}_1$.

If Q = p directions are constructed, then a solution equivalent to least squares solution would then be observed while using Q<p directions produced reduced regression model.

Analysis and Results

MLR Analysis:

Building an MLR model on all the features, we observe the R-squared value for training data as 0.975 while for test data it came out to be 0.930. This raises an issue of overfitting. The algorithm was again executed on the selected features (height, engine-size, wheel-base, horsepower) obtained by the application of RFE, it was observed that the R-squared value decreased to 0.759.

Although, it can be seen from Figure 2 that wheelbase and engine-size are positively correlated with a correlation value of 0.60. This implies that there exist correlated variables which persuaded us to look for other regression approaches.

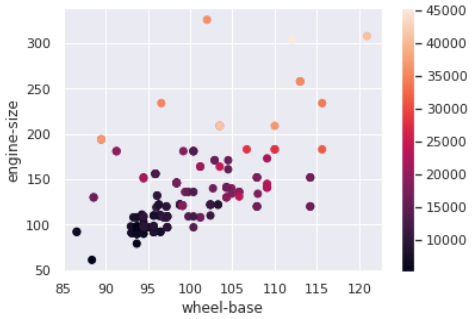

Figure 2. Engine size vs Wheelbase

PCA/PCR Analysis:

The loading weights obtained from the first PC were 0.26, 0.55, 0.59, and 0.53 for height, wheelbase, engine size, and horsepower respectively. The contradiction of the MLR model was then solved by the PCR built using the selected features. Both engine size and wheelbase had their regression coefficient values to be positive. The first component and second component were enough to explain the 91% of the overall variability, hence the regression was done between target price and PC1 and PC2. From figure 3, it can be clearly observed that a regression model built on the first two components gave an R-squared value of 0.746.

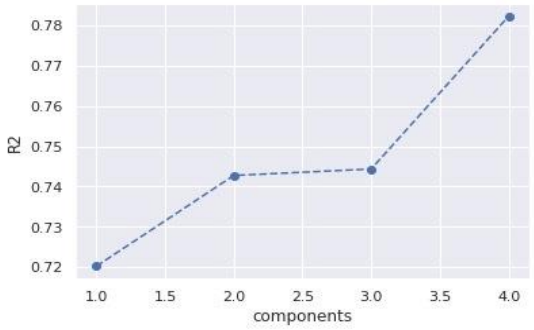

Figure 3. R2 vs principal components

PLSR Analysis:

Applying PLSR model on the selected features with the target variable, it was noticed that the R-squared value slightly increased 0.759 which was noticed to be like that observed using MLR (post feature selection). It can be observed from figure 4, that the R-squared value was stabilized after four partial least square components.

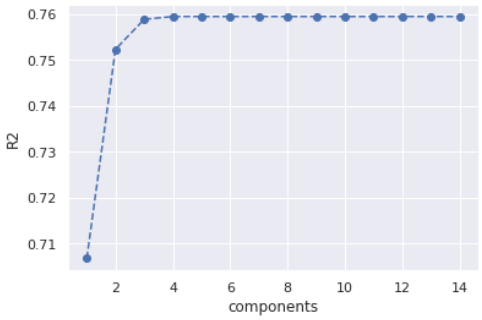

Figure 4. R2 vs PL components

PCR vs PLSR Analysis:

Figure 5 shows the projected data on the first PCA component and on the first PLS component. The difference between them is that the test data of predictors is transformed by using PCA and PLS methods. Further their prediction is plotted with their respective mentioned algorithms. It can be clearly seen that the regression line in PLSR is fitted well with the data set

which cannot be observed in the PCR plot. Hence, PLSR is a promising method as compared to PCR.

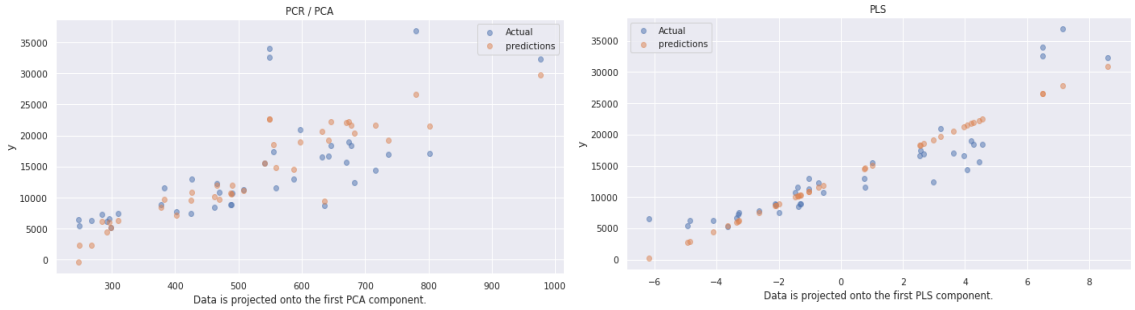

Figure 5. PCR vs PLS

Conclusion

The hypothesis of this study was therefore rejected as PLSR and MLR were observed to perform better than PCR. Though there is a small but significant improvement in the performance accuracy of PLSR as compared to PCR. There are a few limitations to this study. First being that analyzing decision tree beyond depth three is critical. The use of categorical variables in PCA/PCR and PLSR is an issue. It is suggested to try using random forest and support vector in RFE to get another set of significant features. Other researchers can make use of cost-complexity pruning in decision tree. To deal with categorical response variable, it could be a better option to use PLS-Discriminant Analysis (PLS-DA) a variant of PLS.

References

G. James, D. Witten, T. Hastie, and R. Tibshirani, An introduction to statistical learning: With applications in R, 2013th ed. New York, NY: Springer, 2013.

Mehmood, T., Sæbø, S., & Liland, K. H. (2020). Comparison of variable selection methods in partial least squares regression. Journal of Chemometrics, 34(6), e3226.

Rong, M., Shi, H., Song, B., & Tao, Y. (2021). Multi-block dynamic weighted principal component regression strategy for dynamic plant-wide process monitoring. Measurement, 183, 109705.

Pedregosa, F., Varoquaux, Ga"el, Gramfort, A., Michel, V., Thirion, B., Grisel, O., … others. (2011). Scikit-learn: Machine learning in Python. Journal of Machine Learning Research, 12(Oct), 2825–2830

Trevor Hastie, Robert Tibshirani and Jerome Friedman, The Elements of Statistical Learning, 2nd edition, Springer, 2008.

Code : https://colab.research.google.com/drive/1FJmTVn-YiXWO6gTyAAGFlOtIXuNwfBl1?usp=sharing

Appendix

```
 # -*- coding: utf-8 -*-
"""Project_Linear_Model_5320.ipynb
Automatically generated by Colaboratory.
Original file is located at
https://colab.research.google.com/drive/1FJmTVn-YiXWO6gTyAAGFlOtIXuNwfBl1
"""
#Data Pre-processing
# Importing Libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import sklearn
from google.colab import files
import io
import plotly.express as px
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.pipeline import make_pipeline
import sklearn.preprocessing as pp
from sklearn import linear_model
from sklearn import metrics
import seaborn as sns
from sklearn.linear_model import LinearRegression
from statsmodels.api import OLS
from sklearn.ensemble import RandomForestRegressor
from sklearn.tree import DecisionTreeRegressor
import sklearn.tree as tree
from sklearn.feature_selection import RFE
from sklearn.decomposition import PCA
from sklearn.cross_decomposition import PLSRegression
np.set_printoptions(threshold=np.inf)
np.set_printoptions(linewidth=np.inf)
pd.set_option('display.max_rows', 500)
pd.set_option('display.max_columns', 500)
pd.set_option('display.max_seq_items', 500)
file = files.upload()
df = pd.read_csv(io.StringIO(file['auto_clean.csv'].decode('latin-1')))
df_raw = df
df.head()
df.describe(include= "all")
```

```
# Looking the columns of the data
df.columns
df.info()
#number of rows and columns in the data
df.shape
df = pd.get_dummies(df)
features_name=df.columns.to_list()
df.dropna(inplace=True)# group by make/company then average
df.shape
x = df.drop(['price'],axis=1)
y = df['price']
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2, random_state=1)
"""#Multiple Linear Regression"""
reg = LinearRegression()
reg.fit(x_train, y_train)
y_pred_train = reg.predict(x_train)
print("slope:", reg.coef_[0])
print("intercept", reg.intercept_)
# Algorithm Performance
y_pred = reg.predict(x_test)
print("r2 test: ", metrics.r2_score(y_test,y_pred))
print("MSE test: ", metrics.mean_squared_error(y_test,y_pred))
print("r2 train: ", metrics.r2_score(y_train,y_pred_train))
print("MSE train:", metrics.mean_squared_error(y_train,y_pred_train))
print("coef:",reg.coef_)
plt.scatter(df['wheel-base'],df['engine-size'], c= df['price'])
plt.xlabel("wheel-base")
plt.ylabel("engine-size")
plt.colorbar()
slctd_features = ['wheel-base', 'curb-weight', 'engine-size', 'horsepower', 'engine-type_ohc']
df1 = df[['width','horsepower','engine-size' , 'stroke']]
corrmatrix = df1.corr()
```

```
sns.heatmap(corrmatrix, annot=True)
sns.set(rc={'figure.facecolor': 'white'})
plt.show()
"""#RFE"""
input=df.describe().columns.to_list()# saving all the columns
var=df[input]
var=var.drop(['price'],axis=1)# taking all predictors.
x=var.values
y=df['price'].values # defining target
#Standardizing
X_train_std = pp.StandardScaler().fit_transform(x_train)
X_test_std = pp.StandardScaler().fit_transform(x_test)
regr_rfe = DecisionTreeRegressor(criterion='mse',max_depth=2)
regr_rfe.fit(x_train, y_train)
plt.figure( figsize=(20,10) )
_ = tree.plot_tree( regr_rfe,
feature_names = features_name,
#class_names = Class_names,
filled = True)
plt.savefig('tree.png')
forest=DecisionTreeRegressor()
_ = forest.fit(X_train_std, y_train)
forest.score(X_test_std, y_test)
#Listing the best features.
import random
random.seed(10)
rfe = RFE(estimator=DecisionTreeRegressor(),#RandomForestRegressor(),
n_features_to_select=5, step=5)
_ = rfe.fit(X_train_std, y_train)
#print(var.columns[rfe.support_])
slctd_features=var.columns[rfe.support_].to_list()# BMW is an outlier
slctd_features = ['height', 'wheel-base', 'engine-size', 'horsepower', 'engine-type_ohc']
slctd_features
if "engine-type_ohc" in slctd_features: slctd_features.remove("engine-type_ohc")
x_slctd = df[slctd_features]
#from sklearn.model_selection import train_test_split
```

```
x_train_slctd, x_test_slctd, y_train, y_test = train_test_split(x_slctd, y, test_size=0.2,
random_state=1)
X_train_std_slctd = pp.StandardScaler().fit_transform(x_train_slctd)
X_test_std_slctd = pp.StandardScaler().fit_transform(x_test_slctd)
#Linear regression with all the selected standardized features except categorical variable
reg1 = LinearRegression()
reg1.fit(X_train_std_slctd, y_train)
y_pred1 = reg1.predict(X_test_std_slctd)
y_predtrain1 = reg1.predict(X_train_std_slctd)
print("r2 train: ", metrics.r2_score(y_train,y_predtrain1))
print("slope:", reg1.coef_[0])
print("intercept", reg1.intercept_)
# Algorithm Performance
print("r2 test: ", metrics.r2_score(y_test,y_pred1))
print("mean square error: ", metrics.mean_squared_error(y_test,y_pred1))
#Output suggets that the categorical variable made difference to the output, still there is a
difference, slope and mse reduced very lightly.
"""#PCA"""
pca = PCA()
x_train_pca_ = pca.fit_transform(X_train_std_slctd)
x_test_pca_ = pca.transform(X_test_std_slctd)
print(pca.components_)
pca.explained_variance_ratio_
len(pca.components_)
x_train_max_var = x_train_pca_[:,0:2]
x_test_max_var = x_test_pca_[:,0:2]
labels = {
str(i): f"PC {i+1} ({var:.1f}%)"
for i, var in enumerate(pca.explained_variance_ratio_ * 100)
}
fig = px.scatter_matrix(
x_train_max_var,
labels=labels,
dimensions=range(2)
)
fig.show()
```

```
reg_all_pca = LinearRegression()
reg_all_pca.fit(x_train_pca_[:,0:2], y_train)
print("slope:", reg_all_pca.coef_[0])
print("intercept", reg_all_pca.intercept_)
y_pred_all_pca = reg_all_pca.predict(x_test_pca_[:,0:2])
print("r2: ", metrics.r2_score(y_test,y_pred_all_pca))
print("mean square error: ", metrics.mean_squared_error(y_test,y_pred_all_pca))
#Linear regression between raw variables and pc1.
reg_all_new = LinearRegression()
reg_all_new.fit(X_train_std_slctd, x_train_pca_[:,0:1])
print(reg_all_new.coef_)
plt.scatter(reg_all_new.coef_,pca.components_[0],c='purple')
x=[0,1]
y=[0,1]
plt.plot(x,y,c="purple")
"""#PLS"""
pls = PLSRegression(n_components=4)
pls.fit(X_train_std_slctd, y_train)
y_pred = pls.predict(X_test_std_slctd)
print("r2: ", metrics.r2_score(y_test,y_pred))
print("mean square error: ", metrics.mean_squared_error(y_test,y_pred))
n=list(range(1,15))
r=[]
mse=[]
for i in n:
pls = PLSRegression(n_components=i)
pls.fit(X_train_std_slctd, y_train)
y_pred = pls.predict(X_test_std_slctd)
r.append(metrics.r2_score(y_test,y_pred))
mse.append(metrics.mean_squared_error(y_test,y_pred))
fig = plt.figure()
plt.scatter(n,mse)
plt.plot(n, mse, '--')
fig = plt.figure()
plt.plot(n, r, '--')
plt.scatter(n,r)
plt.xlabel("components")
plt.ylabel("R2")
```

```
#plt.plot(n, r, '--')
std = StandardScaler()
n=list(range(1,5))
rsquare=[]
for i in n:
pcr = make_pipeline(StandardScaler(), PCA(n_components=i), LinearRegression())
pcr.fit(x_train, y_train)
rsquare.append(pcr.score(x_test, y_test))
fig = plt.figure()
plt.plot(n, rsquare, '--')
plt.scatter(n,rsquare)
plt.xlabel("components")
plt.ylabel("R2")
"""PCR and PLS Comparison"""
#from sklearn.pipeline import make_pipeline
pcr = make_pipeline(StandardScaler(), PCA(n_components=1), LinearRegression())
pcr.fit(x_train, y_train)
pca = pcr.named_steps["pca"]
pls = PLSRegression(n_components=1)
pls.fit(x_train, y_train)
fig, axes = plt.subplots(2, 1, figsize=(10, 10))
axes[0].scatter(pca.transform(x_test), y_test, alpha=0.5, label="Actual")
axes[0].scatter(pca.transform(x_test), pcr.predict(x_test), alpha=0.5, label="predictions")
axes[0].set(xlabel="Data is projected onto the first PCA component.", ylabel="y", title="PCR / PCA")
axes[0].legend()
axes[1].scatter(pls.transform(x_test), y_test, alpha=0.5, label="Actual")
axes[1].scatter(pls.transform(x_test), pls.predict(x_test), alpha=0.5, label="predictions")
axes[1].set(xlabel="Data is projected onto the first PLS component.", ylabel="y", title="PLS")
axes[1].legend()
plt.tight_layout()
plt.show()
```