MTH 522: Advanced Mathematical Statistics

Building and Validating a Linear Model to Predict Baby Birthweight

04/02/2023

Issues

This report details the construction of a linear model that can be used to predict the birthweight of a baby based on various statistical attributes of the mother. The model is constructed using variables such as Gestation, Age, Height, Weight, and smoking habits of the mother. The focus of this report is to determine the test error of the prediction model. Test error refers to the error that arises when the model is provided with data that it has never encountered before and is asked to predict the baby's weight. Although calculating this error can be challenging and lead to some issues, there are three validation methods that can be used to refine the test error estimate.

Findings

Upon creating a linear model for prediction and testing it through various validation methods including a validation set, leave-one-out cross-validation, and k-fold cross-validation, we were able to identify several key findings. The first finding was that certain variables from the mother were significant to the model and had the greatest impact on accurately predicting the baby's birthweight. These significant variables included the mother's height, weight, and smoking habits. In some cases, gestation also played a role in determining birthweight. The prediction accuracy of the model was calculated using the validation set, leave-one-out cross validation, and k-fold cross validation techniques. The validation set's MSE was 261.65, indicates model's predictions are inaccurate.

Appendix A: Methods

The study utilized a dataset consisting of six variables, namely Gestation, Age, Height, Weight, Smoke (0 or 1 representing non-smoker or smoker), and Birthweight of the child. The primary objective of the study was to predict the birthweight of the child based on the statistical features of the mother. Before beginning the analysis, an empty row and unrealistic values in the dataset were removed, such as numbers like 999, 99, and 9 for Gestation, Age, Height, Weight, and Smoke, respectively. The summary statistics of all variables in the dataset were analysed.

Three validation methods were considered for the developed model: validation set, leave-one-out cross-validation, and k-fold cross-validation. To begin with the validation set approach, the dataset was divided into a training set and a

validation/test set in a 50:50 ratio, and a separate variable for the original target (Birthweight) values was kept. Subsequently, a multivariate linear model was built on the five variables related to the mother, i.e., Gestation, Age, Height, Weight, and Smoke (0 or 1). The significance of each variable in the prediction was evaluated by examining the model summary. Finally, the QQ plot or quantile plot of the residuals was analysed to determine whether the residuals of the model were normally distributed.

Appendix B: Results

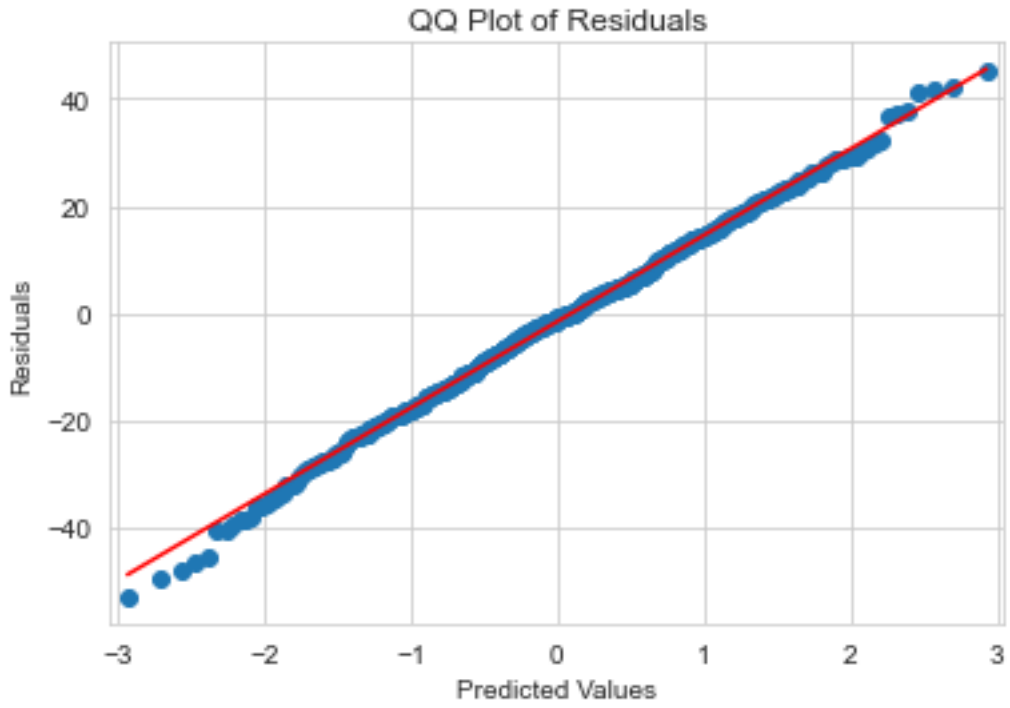
Summary Statistics of the dataset. This summary shows statistics for each of the six variables of the dataset.

	Gestation	Age	Height	Weight	Smoke	Birthweight
count	1174.000000	1174.000000	1174.000000	1174.000000	1174.000000	1174.000000
mean	279.101363	27.228279	64.049404	128.478705	0.390971	119.462521
std	16.010305	5.817839	2.526102	20.734282	0.488176	18.328671
min	148.000000	15.000000	53.000000	87.000000	0.000000	55.000000
25%	272.000000	23.000000	62.000000	114.250000	0.000000	108.000000
50%	280.000000	26.000000	64.000000	125.000000	0.000000	120.000000
75%	288.000000	31.000000	66.000000	139.000000	1.000000	131.000000
max	353.000000	45.000000	72.000000	250.000000	1.000000	176.000000

Below are the summary results from linear model that was built the validation set approach.

	coef	std err	t	P>|t|	[0.025	0.975]
Gestation	0.3549	0.025	14.043	0.000	0.305	0.405
Age	-0.0217	0.080	-0.271	0.786	-0.179	0.136
Height	0.2126	0.128	1.656	0.098	-0.039	0.464
Weight	0.0840	0.025	3.353	0.001	0.035	0.133
Smoke	-8.5229	0.968	-8.808	0.000	-10.421	-6.624

Omnibus:	3.471	Durbin-Watson:	2.054
Prob(Omnibus):	0.176	Jarque-Bera (JB):	3.835
Skew:	0.034	Prob(JB):	0.147
Kurtosis:	3.272	Cond. No.	650.



Quantile plot of the residuals to determine if the residuals follow a normal distribution because all the points will lie on the line

Appendix C: Code

```
#!/usr/bin/env python
# coding: utf-8

# In[1]:


import pandas as pd
import os
import numpy as np
import statsmodels.api as sm
from sklearn.model_selection import train_test_split, LeaveOneOut, KFold
import pylab as py
import matplotlib.pyplot as plt

data = pd.read_excel(os.path.join(os.getcwd(), 'Babies_weight.xlsx'))

data.columns = ['Gestation', 'Age', 'Height', 'Weight', 'Smoke', 'Birthweight']
data.head()
data = data.dropna()

data = data[data["Gestation"] != 999]
data = data[data["Age"] != 99]
data = data[data["Height"] != 99]
```

```
data = data[data["Weight"] != 999]
data = data[data["Smoke"] != 9]
data.describe()

# Split the data into predictor variables (X) and outcome variable (y)
X = data[['Gestation', 'Age', 'Height', 'Weight', 'Smoke']]
y = data['Birthweight']

model = sm.OLS(y, X).fit()
print(model.summary())

# Use validation set method
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.5, random_state=0)

# Fit the model on the training set
model_vs = sm.OLS(y_train, X_train).fit()

# Predict the outcome variable on the test set
y_pred_vs = model_vs.predict(X_test)

# Calculate the mean squared error (MSE) on the test set
mse_vs = np.mean((y_test - y_pred_vs) ** 2)
print('MSE for validation set method:', mse_vs)
rmse_vs = np.sqrt(mse_vs)
print('RMSE for validation set method: ',rmse_vs)

residuals = y_test - y_pred_vs
sm.qqplot(residuals, line='s')
plt.title('QQ Plot of Residuals')
plt.xlabel('Predicted Values')
plt.ylabel('Residuals')
plt.show()

# Use LOOCV
loocv = LeaveOneOut()
loocv.get_n_splits(X)
# Initialize empty arrays to store predictions and actual values
y_pred_loocv = np.zeros(len(y))
y_actual_loocv = np.zeros(len(y))

for train_idx, test_idx in loocv.split(X):
    X_train, X_test = X.iloc[train_idx], X.iloc[test_idx]
    y_train, y_test = y.iloc[train_idx], y.iloc[test_idx]
    model_loocv = sm.OLS(y_train, X_train).fit()
    y_pred_loocv[test_idx] = model_loocv.predict(X_test)
    y_actual_loocv[test_idx] = y_test


mse_loocv = np.mean((y_actual_loocv - y_pred_loocv) ** 2)
print('MSE for LOOCV: ', mse_loocv)
```

```
rmse_loocv = np.sqrt(mse_loocv)
print('RMSE for LOOCV: ', rmse_loocv)

# Use k-fold cross-validation with k=10
kfold = KFold(n_splits=10)

# Initialize empty array to store MSE for each fold
mse_kfold = np.zeros(10)

for i, (train_idx, test_idx) in enumerate(kfold.split(X)):
    X_train, X_test = X.iloc[train_idx], X.iloc[test_idx]
    y_train, y_test = y.iloc[train_idx], y.iloc[test_idx]
    model_kfold = sm.OLS(y_train, X_train).fit()
    y_pred_kfold = model_kfold.predict(X_test)
    mse_kfold[i] = np.mean((y_test - y_pred_kfold) ** 2)

# Calculate the mean MSE over all folds
mean_mse_kfold = np.mean(mse_kfold)
print('Mean MSE for k-fold cross-validation:', mean_mse_kfold)
rmse_kfold = np.sqrt(mean_mse_kfold)
print('RMSE for k-fold cross-validation:', rmse_kfold)
```