Homework 6 - Regression

In this guide, we will be exploring using regression as an intro to artificial intelligence. For this week's assignment, we will be exploring linear regression. We'll be using the data from our soccer database from assignment 4.

Instructions

1. Follow the instructions on how to setup your Python and Jupyter (or VSCode) environment and cloning or downloading our repository. Instructions can be found in the class notes.
2. Import soccer database using pandas.
3. Load the values from the attributes `gk_reflexes` and `gk_handling` from table `Player_Attributes` .
4. Use `gk_reflexes` (as x) and `gk_handling` (as y) as your data.
5. Drop the missing values from these two columns.
6. Scale the dataset using a standard scaler.
7. Split the data into training and testing in a 0.3 ratio (70% training, 30% testing).
8. Apply Linear Regression, Cross-Validation (with 5 splits), Ridge Regularization, and Lasso Regularizations and print the co-relation result of each technique using `r2_score` . All of the functions for this last step are located in sklearn.
9. Answer the questions in the notebook through code.
10. Run the notebook and make sure everything works.
11. Export the notebook as HTML or PDF.
12. Submit the notebook through Canvas.

Remember to fill the missing pieces of code in the provided notebook.

Dataset Overview

The dataset covers information about soccer players in sqlite format. This file is located in the `Datasets` directory of this repository. The file is called `fifa_soccer_dataset.sqlite.gz` . **This is the same file from the previous homework (assignment 4).**

If you haven't decompressed the file, you may need to follow the instructions below to decompress it.

IMPORTANT The database is compressed and needs to be decompressed before use. You can do this by running the following command in your terminal on Linux or MacOS:

```
gunzip Datasets/fifa_soccer_dataset.sqlite.gz
```

If you are using Windows, you can use the following command in your powershell:

```
$sourceFile = "$PWD\Datasets\fifa_soccer_dataset.sqlite.gz"
$destinationFile = "$PWD\Datasets\fifa_soccer_dataset.sqlite"

$inputStream = [System.IO.File]::OpenRead($sourceFile)
$outputStream = [System.IO.File]::Create($destinationFile)
$gzipStream = New-Object System.IO.Compression.GzipStream($inputStream,
[System.IO.Compression.CompressionMode]::Decompress)
$gzipStream.CopyTo($outputStream)

$gzipStream.Close()
$outputStream.Close()
$inputStream.Close()
```

Alternatively, you can extract the file using the GUI of your operating system.

Submission Guidelines

- Submit your completed notebook as a HTML export, or a PDF file.

To export to HTML, if you are on Jupyter, select `File` > `Export Notebook As` > `HTML` .

If you are on VSCode, you can use the `Jupyter: Export to HTML` command.

- Open the command palette (Ctrl+Shift+P or Cmd+Shift+P on Mac).
  - Search for `Jupyter: Export to HTML` .
  - Save the HTML file to your computer and submit it via Canvas.

To begin, we'll need quite a few imports.

In [1]:
```
import pandas as pd
import numpy as np
import sqlite3
from sklearn.model_selection import train_test_split, KFold
```

```python
from sklearn.linear_model import LinearRegression, Ridge, Lasso
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import r2_score
```

We're going to use the soccer data to run regressions. In the cell below, connect to the database.

In [2]:
```python
dataset_path = "/Users/rad/Desktop/Useable Ai/Assignments/Final Submission/HW6/fifa_soccer_dataset.sqlite"
conn = sqlite3.connect(dataset_path)
```

To get started, let's write a query to grab all of the entries from the `Player_Attributes` table, and print the first 5 rows below.

In [3]:
```python
player_attr_df = pd.read_sql("SELECT * FROM Player_Attributes", conn)

# Display the first 5 rows of the table
player_attr_df.head()
```

Out[3]:

	id	player_fifa_api_id	player_api_id	date	overall_rating	potential	preferred_foot	attacking_work_rate	defensive_work_rate
0	1	218353	505942	2016-02-18 00:00:00	67.0	71.0	right	medium	medium
1	2	218353	505942	2015-11-19 00:00:00	67.0	71.0	right	medium	medium
2	3	218353	505942	2015-09-21 00:00:00	62.0	66.0	right	medium	medium
3	4	218353	505942	2015-03-20 00:00:00	61.0	65.0	right	medium	medium
4	5	218353	505942	2007-02-22 00:00:00	61.0	65.0	right	medium	medium

5 rows × 42 columns

We are going to play with two fields today, the `gk_handling` field as the dependent feature and the `gk_reflexes` field as the independent feature. Let's drop some missing values from these two columns as well. They represent the goalkeeping handling and reflexes of a player respectively.

In [4]:
```python
player_attr_df = player_attr_df.dropna(subset=["gk_handling", "gk_reflexes"])
```

In [6]:
```python
player_attr_df[["gk_handling", "gk_reflexes"]].head()
```

Out[6]:

	gk_handling	gk_reflexes
0	11.0	8.0
1	11.0	8.0
2	11.0	8.0
3	10.0	7.0
4	10.0	7.0

Let's store those columns in their own variables for easy reading.

In [7]:
```python
x = player_attr_df[['gk_reflexes']].values
y = player_attr_df[['gk_handling']].values
```

To preform and evaluate our linear regression, we need to split our data into test and training batches. We can do this by using the `train_test_split()` function. In the cell below, use this function and pass it `x` and `y` as the data for it to split. The final parameter `test_size` indicates how big the test batch should be, in this case 30% of the initial dataset inputted.

In [8]:
```python
X_train, X_test, Y_train, Y_test = train_test_split(x, y, test_size=0.3, random_state=42)
```

We can now preform the fitting. Let's call the `fit()` function on our `lm` variable, passing the `X_train` and `Y_train` data as parameters.

In [9]:
```python
lm = LinearRegression()
lm.fit(X_train, Y_train)
```

Out[9]:

```
▾ LinearRegression
LinearRegression()
```

Great! Now we can use the predict funtion to see how the model preforms against our test data set. Call the `predict()` function on `lm` and pass `X_test` as our input parameter. We'll then see the r2 score to see how correlated these values are.

In [10]:

```
Y_predicted = lm.predict(X_test)
rsquared = r2_score(Y_test, Y_predicted)
print("R2 Score: " + str(rsquared))
```

```
R2 Score: 0.9343683631199424
```

These values are pretty correlated! We can also use the `StandarScalar()` to transform our values before fitting our model. In the cell below, call the `StandardScalar()` function and pass `x` and `y` to the `fit_transform()` functions.

In [11]:

```
sc = StandardScaler()

x_scaled = sc.fit_transform(x)
y_scaled = sc.fit_transform(y)
```

Now we can run the model again as we did before. We'll need to split the training and test batches again, then run a new `fit()`. Once fitted, we can again use `predict()` and run a r2 score again.

In [12]:

```
X_train, X_test, Y_train, Y_test = train_test_split(x_scaled, y_scaled, test_size=0.3, random_state=42)
```

In [13]:

```
lm = LinearRegression()
lm.fit(X_train, Y_train)
```

Out[13]:

```
▾ LinearRegression
LinearRegression()
```

In [15]:

```
Y_predicted = lm.predict(X_test)
rsquared = r2_score(Y_test, Y_predicted)
print("R2 Score: " + str(rsquared))
```

```
R2 Score: 0.9343683631199426
```

Implementing various models - LinearRegression(), Ridge(), Lasso() along with K-Fold CrossValidation with 5 splits. Use the unscaled data for this step.

In [16]:

```
# Apply Linear regression, ridge regularization, lasso regularization with cross validation

# Define models
model_lr = LinearRegression()
model_ridge = Ridge(alpha=1.0)
model_lasso = Lasso(alpha=0.1)

# Cross validation
kf = KFold(n_splits=5)
list_r2_score = []


# Split the train set:
for train_index, test_index in kf.split(x):
    X_train, X_test = x[train_index], x[test_index]
    y_train, y_test = y[train_index], y[test_index]
    k_fold_r2 = []
    for model in [model_lr, model_ridge, model_lasso]:
        model.fit(X_train, y_train)
        pred = model.predict(X_test)
        k_fold_r2.append(r2_score(y_test,pred))

    list_r2_score.append(k_fold_r2)

# Show the result - Add Mean and Standard Deviation of the R2-scores
list_r2_score.append(list(np.mean(list_r2_score, axis=0)))
list_r2_score.append(list(np.std(list_r2_score[:-1], axis=0)))


result_r2 = pd.DataFrame(list_r2_score)
result_r2.columns = ['Linear Regression', 'Ridge', 'Lasso']
result_r2.index = ['k1', 'k2', 'k3', 'k4', 'k5', 'average', 'std']

print('The result of r2 scores for k=5 cross validation')
display(result_r2)
```

```
The result of r2 scores for k=5 cross validation
```

	Linear Regression	Ridge	Lasso
k1	0.932690	0.932690	0.932693
k2	0.928483	0.928483	0.928481
k3	0.932024	0.932024	0.932024
k4	0.931913	0.931913	0.931915
k5	0.942354	0.942354	0.942351
average	0.933493	0.933493	0.933493
std	0.004667	0.004667	0.004666

And that's basic linear regression with python. Please turn in this notebook completed with your outputs displayed in html or pdf formats.