Concrete strength prediction model analysis report and project coding part.

- About dataset and its goal.
- Project analysis report.
- Notebook code.
- Flask and html code (frontend and backend)

Dataset & its Goal.

The given dataset contains information on the compressive strength of concrete. The data has nine input variables and one target variable. The input variables are:

cement: the amount of cement (in kg/m^3) in the concrete mixture

blastFurnace: the amount of blast furnace slag (in kg/m^3) in the concrete mixture

flyAsh: the amount of fly ash (in kg/m^3) in the concrete mixture

water: the amount of water (in kg/m^3) in the concrete mixture

superplasticizer: the amount of superplasticizer (in kg/m^3) in the concrete mixture

courseAggregate: the amount of coarse aggregate (in kg/m^3) in the concrete mixture

fineAggregate: the amount of fine aggregate (in kg/m^3) in the concrete mixture

age: the age (in days) of the concrete sample when the compressive strength was measured

The target variable is:

strength: the compressive strength of the concrete (in MPa)

The goal of this dataset is to develop a predictive model that can accurately estimate the compressive strength of concrete based on the given input variables. This model can be useful for engineers and construction companies to optimize the design of concrete mixtures and ensure that the resulting concrete structures meet the required strength specifications.

Concrete Compressive Strength

Data Type: multivariate

Abstract: Concrete is the most important material in civil engineering. The concrete compressive strength is a highly nonlinear function of age and ingredients. These ingredients include cement, blast furnace slag, fly ash, water, superplasticizer, coarse aggregate, and fine aggregate.

Sources:

Original Owner
Vikash singh chauhan
Email- vikashchauhanvv26@gmail.com

Data Characteristics:

The actual concrete compressive strength (MPa) for a given mixture under a specific age (days) was determined from laboratory. Data is in raw form (not scaled).

Summary Statistics:

Number of instances (observations): 1030
Number of Attributes: 9
Attribute breakdown: 8 quantitative input variables, and 1 quantitative output variable
Missing Attribute Values: None

Variable Information:

Given is the variable name, variable type, the measurement unit and a brief description. The concrete compressive strength is the regression problem. The order of this listing corresponds to the order of numerals along the rows of the database.

Name -- Data Type -- Measurement -- Description

Cement (component 1) -- quantitative -- kg in a m3 mixture -- Input Variable
Blast Furnace Slag (component 2) -- quantitative -- kg in a m3 mixture -- Input Variable
Fly Ash (component 3) -- quantitative -- kg in a m3 mixture -- Input Variable
Water (component 4) -- quantitative -- kg in a m3 mixture -- Input Variable
Superplasticizer (component 5) -- quantitative -- kg in a m3 mixture -- Input Variable
Coarse Aggregate (component 6) -- quantitative -- kg in a m3 mixture -- Input Variable
Fine Aggregate (component 7) -- quantitative -- kg in a m3 mixture -- Input Variable
Age -- quantitative -- Day (1~365) -- Input Variable
Concrete compressive strength -- quantitative -- MPa -- Output Variable

---

Past Usage:

Main

1. I-Cheng Yeh, "Modeling of strength of high performance concrete using artificial neural networks," Cement and Concrete Research, Vol. 28, No. 12, pp. 1797-1808 (1998).

Others

2. I-Cheng Yeh, "Modeling Concrete Strength with Augment-Neuron Networks," J. of Materials in Civil Engineering, ASCE, Vol. 10, No. 4, pp. 263-268 (1998).

3. I-Cheng Yeh, "Design of High Performance Concrete Mixture Using Neural Networks," J. of Computing in Civil Engineering, ASCE, Vol. 13, No. 1, pp. 36-42 (1999).

4. I-Cheng Yeh, "Prediction of Strength of Fly Ash and Slag Concrete By The Use of Artificial Neural Networks," Journal of the Chinese Institute of Civil and Hydraulic Engineering, Vol. 15, No. 4, pp. 659-663 (2003).

5. I-Cheng Yeh, "A mix Proportioning Methodology for Fly Ash and Slag Concrete Using Artificial Neural Networks," Chung Hua Journal of Science and Engineering, Vol. 1, No. 1, pp. 77-84 (2003).

6. Yeh, I-Cheng, "Analysis of strength of concrete using design of experiments and neural networks,": Journal of Materials in Civil Engineering, ASCE, Vol.18, No.4, pp.597-604 ?2006?.

Acknowledgements, Copyright Information, and Availability:

Project analysis report:

Project Analysis Report: Concrete Compressive Strength Prediction

Introduction

This report presents an analysis of the Concrete Compressive Strength data set, which contains information about various components of concrete and their compressive strength. The goal of this analysis is to build a model that can predict the compressive strength of concrete based on its components.

Data Exploration

The Concrete Compressive Strength data set consists of 1030 samples with 9 features:

cement (component 1)(kg in a m3 mixture)

blast furnace slag (component 2)(kg in a m3 mixture)

fly ash (component 3)(kg in a m3 mixture)

water (component 4)(kg in a m3 mixture)

superplasticizer (component 5)(kg in a m3 mixture)

coarse aggregate (component 6)(kg in a m3 mixture)

fine aggregate (component 7)(kg in a m3 mixture)

age (day)

strength (MPa)

The first step in data exploration was to load the data set and examine its structure. The data set was loaded using the pandas library in Python, and its structure was examined using various pandas functions, such as head(), describe(), info(), and shape().

Data Cleaning

The data set was cleaned by renaming some of the feature columns to make them more readable, and by checking for and dropping any duplicate rows.

Data Preprocessing

The data set was preprocessed by performing several steps, such as handling missing values, scaling the data, and splitting the data into training and testing sets.

Handling Missing Values: There were no missing values in the data set, so no imputation was necessary.

Scaling the Data: The data was scaled using the StandardScaler function from the sklearn.preprocessing library to ensure that all features were on the same scale and to prevent any single feature from dominating the model.

Splitting the Data: The data was split into training and testing sets using the train_test_split() function from the sklearn.model_selection library.

Model Selection

Several models were trained and tested on the data set, including Linear Regression, Lasso Regression, Ridge Regression, Support Vector Regression, Decision Tree Regression, Random Forest Regression, and XGBoost Regression. The model with the best performance was selected based on various evaluation metrics, such as mean squared error (MSE), mean absolute error (MAE), and R-squared score.

Model Evaluation

The selected model was evaluated using the test data set, and its performance was analyzed using various evaluation metrics. The results were visualized using various plots, such as scatter plots and residual plots.

Conclusion

In conclusion, this project aimed to build a model that could predict the compressive strength of concrete based on its components. The data set was explored, cleaned, preprocessed, and used to train and test various machine learning models. The best model was selected based on its performance on the test data set, and its performance was analyzed using various evaluation metrics. The final model can be used to predict the compressive strength of concrete and can be helpful in the construction industry.

Note Book Code:

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from scipy import stats
```

laoding Data

```
[96]
concrete = pd.read_excel("Concrete_Data.xls")
[97]
concrete.head()
```

renaming all features

```
[98]
concrete.head()
[99]
concrete.columns = ['cement','blastFurnace','flyAsh','water','superplasticizer','cour
seAggregate','fineaggregate','age','strength']
[100]
concrete
```

Ask Six Questions

```
[101]
concrete.shape
(1030, 9)
[102]
concrete.isnull().sum()
cement              0
blastFurnace        0
flyAsh              0
water               0
superplasticizer    0
courseAggregate     0
fineaggregate       0
age                 0
strength            0
dtype: int64
[103]
concrete.duplicated().sum()
25
[104]
concrete.drop_duplicates()
[105]
concrete.info()
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1030 entries, 0 to 1029
Data columns (total 9 columns):
```

```
 #   Column            Non-Null Count  Dtype  
---  ------            --------------  -----  
 0   cement            1030 non-null   float64
 1   blastFurnace      1030 non-null   float64
 2   flyAsh            1030 non-null   float64
 3   water             1030 non-null   float64
 4   superplasticizer  1030 non-null   float64
 5   courseAggregate   1030 non-null   float64
 6   fineaggregate     1030 non-null   float64
 7   age               1030 non-null   int64  
 8   strength          1030 non-null   float64
dtypes: float64(8), int64(1)
memory usage: 72.5 KB
```

```
[106]
concrete.describe()
```

EDA

```
[107]
corr = concrete.corr()
```

```
[108]
sns.heatmap(corr,annot=True,cbar=True, cmap='coolwarm')
<AxesSubplot:>
```

Train test split

```
[109]
# Split the data into features and target
from sklearn.model_selection import train_test_split
X = concrete.drop("strength", axis=1)
y = concrete["strength"]

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state
=42)
```

plotting the distribution without any trasnformation

```
[110]
# input_features (input)
for col in X_train.columns:
    plt.figure(figsize=(14,4))
    plt.subplot(121)
    sns.distplot(X_train[col])
    plt.title(col)
```

```
#     # QQ plot
    plt.subplot(122)
    stats.probplot(X_train[col], dist='norm',plot=plt)
    plt.title(col)
    plt.show()
```

Applying PowerTransformer

What is powertransfomer in machine learning?

Ans:

Power transformation is a family of data transformations that are applied to make the data more Gaussian-like or more symmetric. This can be useful for certain statistical methods that assume normally distributed data, such as linear regression.

Types:

Yeo-Johnson & Box-Cox

is a generalization of Box-Cox that allows for transformation of both positive and negative values. PowerTransformer can be used as a preprocessing step in a machine learning pipeline to improve the performance of models that are sensitive to the distribution of the data.

[111]
```
from sklearn.preprocessing import PowerTransformer
pt = PowerTransformer(method='box-cox') # by heo johnson
```
[112]
```
X_train_transformed = pt.fit_transform(X_train+0.000001)
X_test_transoformed = pt.transform(X_test+0.000001)
```

Again Distribution

[113]
```
# input_features (input)
X_train_transformed = pd.DataFrame(X_train_transformed, columns=X_train.columns)

for col in X_train_transformed.columns:
    plt.figure(figsize=(14,4))
    plt.subplot(121)
    sns.distplot(X_train[col])
    plt.title(col)
```

```
#     # QQ plot
    plt.subplot(122)
    sns.distplot(X_train_transformed[col])
    plt.title(col)
    plt.show()
C
```

Scaling (Standard Scalar)

```
[114]
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()

# Fit on training set only.
scaler.fit(X_train)

# Apply transform to both the training set and the test set.
X_train_transformed = scaler.transform(X_train)
X_test_transoformed = scaler.transform(X_test)
```

Training Models:

```
[115]
from sklearn.linear_model import LinearRegression, Ridge, Lasso
from sklearn.model_selection import train_test_split, cross_val_score
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.ensemble import RandomForestRegressor
from xgboost import XGBRegressor

from sklearn.metrics import r2_score


# Define the regression models
models = {
    'lin_reg': LinearRegression(),
    'ridge': Ridge(),
    'lasso': Lasso(),
    'rf_reg': RandomForestRegressor(n_estimators=100, random_state=42),
    'xgb_reg': XGBRegressor(),

}
```

```
for name ,model in models.items():
    model.fit(X_train_transformed,y_train)
    y_pred = model.predict(X_test_transoformed)

    print(f"{name} : {r2_score(y_test,y_pred)}")

lin_reg : 0.6275416055429018
ridge : 0.6275641808582493
lasso : 0.5638889487150622
rf_reg : 0.8818941970097077
xgb_reg : 0.9043842687098029
```

selecting best model

```
[116]
xgb = XGBRegressor()
xgb.fit(X_train_transformed,y_train)
y_pred = xgb.predict(X_test_transoformed)
r2_score(y_test,y_pred)
0.9043842687098029
```

prediction Model

```
[118]
def predicion_system(cem,blastf,flyas,water,superplaster,courseagg,fineagg,age):
    features = np.array([[cem,blastf,flyas,water,superplaster,courseagg,fineagg,age]]
)
    prediction = xgb.predict(features).reshape(1,-1)

    return prediction[0]
[120]
X_train
[122]
cem = 158.60
blastf = 148.90
flyas = 116.00
water = 175.10
superplaster = 15.00
courseagg = 953.3
fineagg = 719.70
age = 28

prediction = predicion_system(cem,blastf,flyas,water,superplaster,courseagg,fineagg,a
ge)
print("strength is : ",prediction)
strength is :  [55.434124]

[123]
```

```
import pickle
pickle.dump(xgb,open('model.pkl','wb'))
```

Flask Code:

```
from flask import Flask, request, render_template
import numpy as np
import pandas as pd
import pickle

# loading model
model = pickle.load(open('model.pkl','rb'))

# creating app
app = Flask(__name__)

@app.route('/')
def index():
    return render_template('index.html')

@app.route('/predict', methods=['POST'])
def predict():
    #   cement    blastFurnace    flyAsh    water    superplasticizer
courseAggregate    fineaggregate    age
    cement = float(request.form['cement'])
    blastFurnace = float(request.form['blastFurnace'])
    flyAsh = float(request.form['flyAsh'])
    water = float(request.form['water'])
    superplasticizer = float(request.form['superplasticizer'])
    courseAggregate = float(request.form['courseAggregate'])
    fineaggregate = float(request.form['fineaggregate'])
    age = int(request.form['age'])

    # transform input features
    features = np.array([cement, blastFurnace, flyAsh, water,
superplasticizer, courseAggregate, fineaggregate, age]).reshape(1, -1)
    prediction = model.predict(features).reshape(1, -1)

    return render_template('index.html', strength=prediction[0][0])

# python main
if __name__ == "__main__":
    app.run(debug=True)
```

Front End:

```
<!doctype html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <title>Concrete Strength Prediction System</title>
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-
alpha3/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-
KK94CHFLLe+nY2dmCWGMq91rCGa5gtU4mk92HdvYe+M/SXH301p5ILy+dN9+nJOZ"
crossorigin="anonymous">
  </head>
  <style>
      .container{
      background:black;
      color: white;
      border-radius:15px;
      padding-bottom:20px;
      }
      body{
      background:light;
      }
      .btn{
      margin-left:38%;
      margin-top:20px;
      }
      .img {
    width: 50%;
    margin-left: 26%;
    margin-right: 93px;
}
  </style>


  <body>
  <h1 class="text-center">Concrete Strength Prediction</h1>
      <img class="img" src="{{url_for('static',filename='img.jpg')}}" alt="">



 <div class="container my-3 mt-3">
  <h1 class="text-center text-success">Concrete Strength Prediction
System</h1>

  <form action="/predict" method="post">
    <div class="row">
      <div class="col">
        <div class="form-group">
```

```
          <label for="cement">Cement</label>
          <input type="number" class="form-control" id="cement" name="cement"
value="0.00" step="any">
        </div>
      </div>
      <div class="col">
        <div class="form-group">
          <label for="blastFurnace">blastFurnace</label>
          <input type="number" class="form-control" id="blastFurnace"
name="blastFurnace" value="0.00" step="any">
        </div>
      </div>
      <div class="col">
        <div class="form-group">
          <label for="flyAsh">flyAsh</label>
          <input type="number" class="form-control" id="flyAsh" name="flyAsh"
value="0.00" step="any">
        </div>
      </div>
    </div>

    <div class="row">
      <div class="col">
        <div class="form-group">
          <label for="water">water</label>
          <input type="number" class="form-control" id="water" name="water"
value="0.00" step="any">
        </div>
      </div>
      <div class="col">
        <div class="form-group">
          <label for="superplasticizer">superplasticizer</label>
          <input type="number" class="form-control" id="superplasticizer"
name="superplasticizer" value="0.00" step="any">
        </div>
      </div>
      <div class="col">
        <div class="form-group">
          <label for="courseAggregate">courseAggregate</label>
          <input type="number" class="form-control" id="courseAggregate"
name="courseAggregate" value="0.00" step="any">
        </div>
      </div>
    </div>

    <div class="row">
      <div class="col">
        <div class="form-group">
          <label for="fineaggregate">fineaggregate</label>
          <input type="number" class="form-control" id="fineaggregate"
name="fineaggregate" value="0.00" step="any">
        </div>
      </div>
      <div class="col">
        <div class="form-group">
          <label for="age">age</label>
          <input type="number" class="form-control" id="age" name="age"
```

```
step="any">
        </div>
      </div>
    </div>

    <button type="submit" class="btn btn-primary btn-lg">get
strength</button>
  </form>
</div>


      {% if strength %}
        <h5 class="text-center">Concrete</h5>
        <p class="text-center">Strength: {{strength}}</p>
      {% endif %}


    <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-
alpha3/dist/js/bootstrap.bundle.min.js" integrity="sha384-
ENjdO4Dr2bkBIFxQpeoTz1HIcje39Wm4jDKdf19U8gI4ddQ3GYNS7NTKfAdVQSZe"
crossorigin="anonymous"></script>
  </body>
</html>
```