Data Glacier

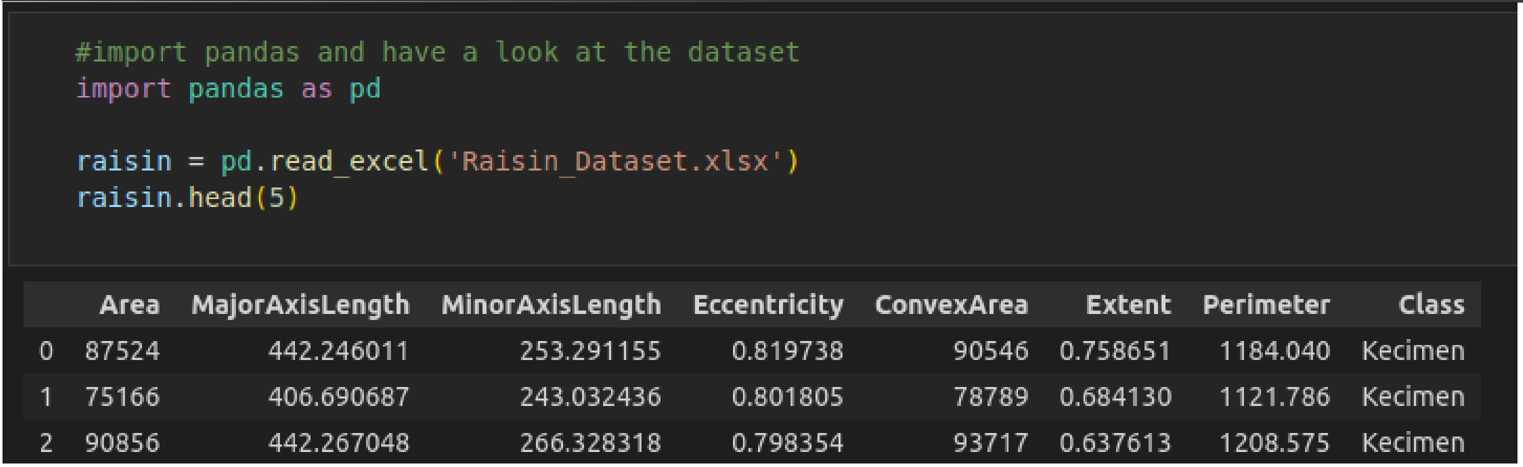
Your Deep Learning Partner

Deployment on Flask

Raisin type prediction app

Name - Margarita Prokhorovich,
Batch code - LISUM09,
Submission date – 25 May, 2022,
Submitted to – Data Glacier
Email: marusya15071240@gmail.com

Model creation

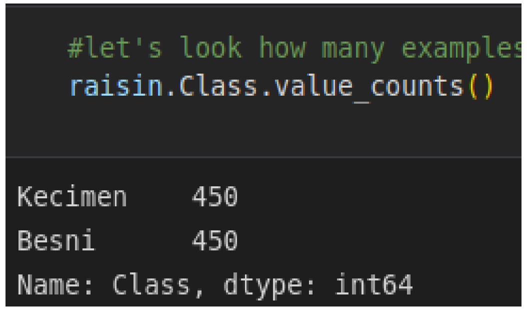


```
#import pandas and have a look at the dataset
import pandas as pd

raisin = pd.read_excel('Raisin_Dataset.xlsx')
raisin.head(5)
```

	Area	MajorAxisLength	MinorAxisLength	Eccentricity	ConvexArea	Extent	Perimeter	Class
0	87524	442.246011	253.291155	0.819738	90546	0.758651	1184.040	Kecimen
1	75166	406.690687	243.032436	0.801805	78789	0.684130	1121.786	Kecimen
2	90856	442.267048	266.328318	0.798354	93717	0.637613	1208.575	Kecimen

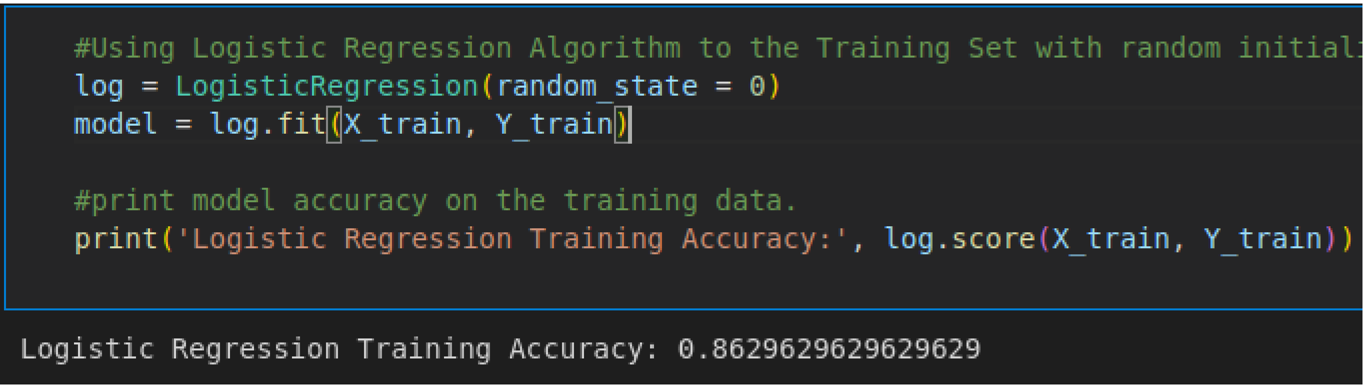
I created a model that predicts raisin type based on different morphological features extracted based on image processing (binary classification problem). I took a data set from Kaggle. More detailed description and the model itself is provided in model.ipynb file. Here I display key aspects.

Reading and displaying the data set. There is equal number of examples in both classes.



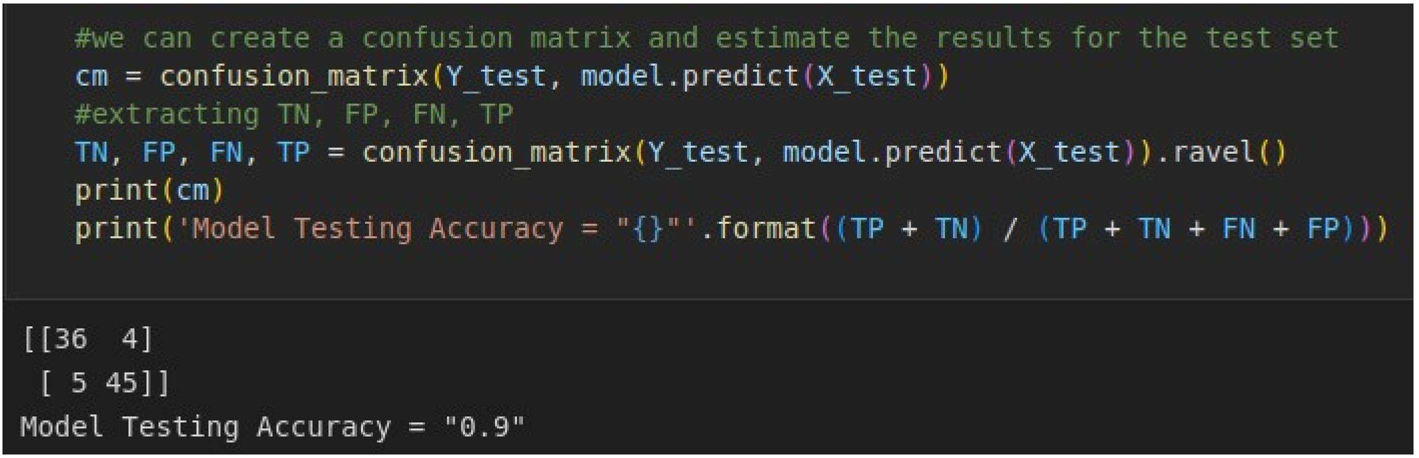
```
#let's look how many examples
raisin.Class.value_counts()
```

```
Kecimen    450
Besni      450
Name: Class, dtype: int64
```



```
#Using Logistic Regression Algorithm to the Training Set with random initial
log = LogisticRegression(random_state = 0)
model = log.fit(X_train, Y_train)

#print model accuracy on the training data.
print('Logistic Regression Training Accuracy:', log.score(X_train, Y_train))
```

```
Logistic Regression Training Accuracy: 0.8629629629629629
```



```
#we can create a confusion matrix and estimate the results for the test set
cm = confusion_matrix(Y_test, model.predict(X_test))
#extracting TN, FP, FN, TP
TN, FP, FN, TP = confusion_matrix(Y_test, model.predict(X_test)).ravel()
print(cm)
print('Model Testing Accuracy = "{}"'.format((TP + TN) / (TP + TN + FN + FP)))
```

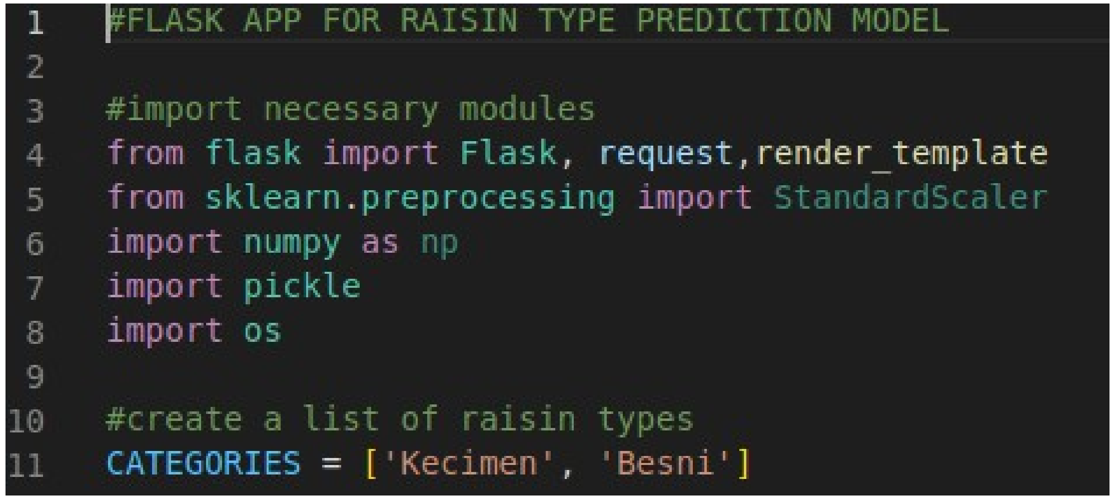
```
[[36  4]
 [ 5 45]]
Model Testing Accuracy = "0.9"
```

Training model on the training and test set gives 86,3% and 90% accuracy respectively.

Flask App creation

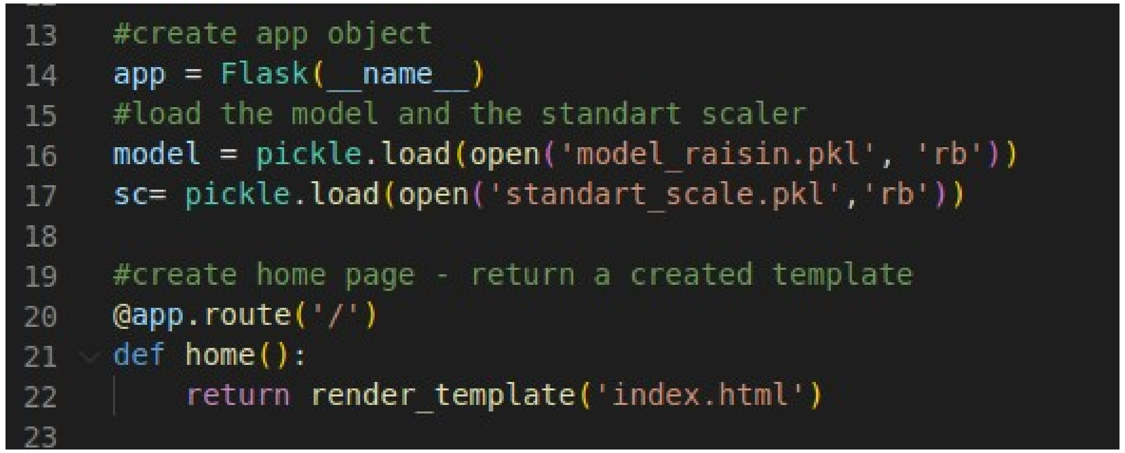
Next step is to create a Flask app itself. Main stages are: import necessary packages, create app object, create pages using decorators, unload the model and make prediction, launch the app. The code is stored in the app.py file.

The first part



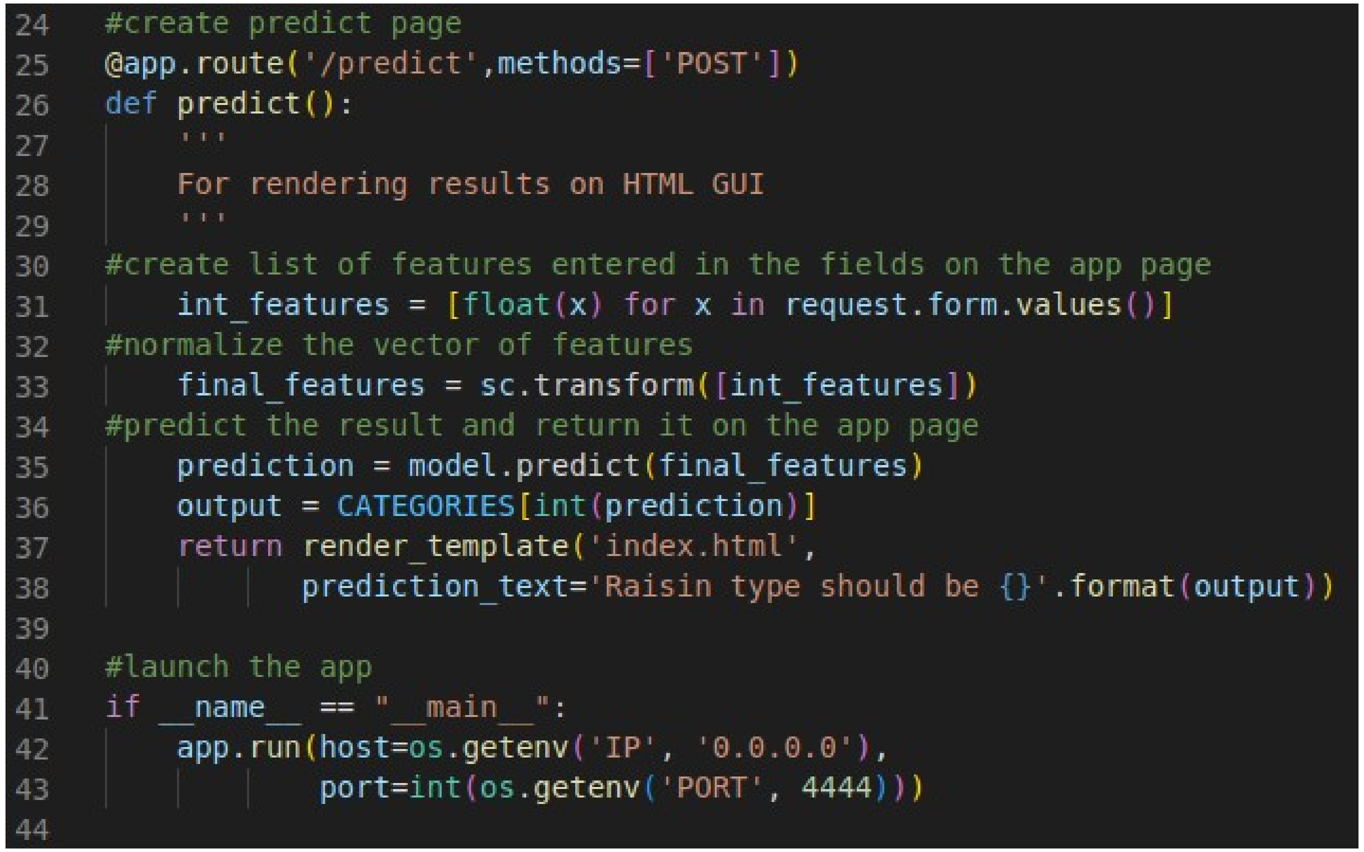
```python
#FLASK APP FOR RAISIN TYPE PREDICTION MODEL

#import necessary modules
from flask import Flask, request,render_template
from sklearn.preprocessing import StandardScaler
import numpy as np
import pickle
import os

#create a list of raisin types
CATEGORIES = ['Kecimen', 'Besni']
```

The second part



```python
#create app object
app = Flask(__name__)
#load the model and the standart scaler
model = pickle.load(open('model_raisin.pkl', 'rb'))
sc= pickle.load(open('standart_scale.pkl','rb'))

#create home page - return a created template
@app.route('/')
def home():
    return render_template('index.html')

```

The third part



```python
#create predict page
@app.route('/predict',methods=['POST'])
def predict():
    '''
    For rendering results on HTML GUI
    '''
#create list of features entered in the fields on the app page
    int_features = [float(x) for x in request.form.values()]
#normalize the vector of features
    final_features = sc.transform([int_features])
#predict the result and return it on the app page
    prediction = model.predict(final_features)
    output = CATEGORIES[int(prediction)]
    return render_template('index.html',
                prediction_text='Raisin type should be {}'.format(output))

#launch the app
if __name__ == "__main__":
    app.run(host=os.getenv('IP', '0.0.0.0'),
            port=int(os.getenv('PORT', 4444)))

```

Creating templates and styles

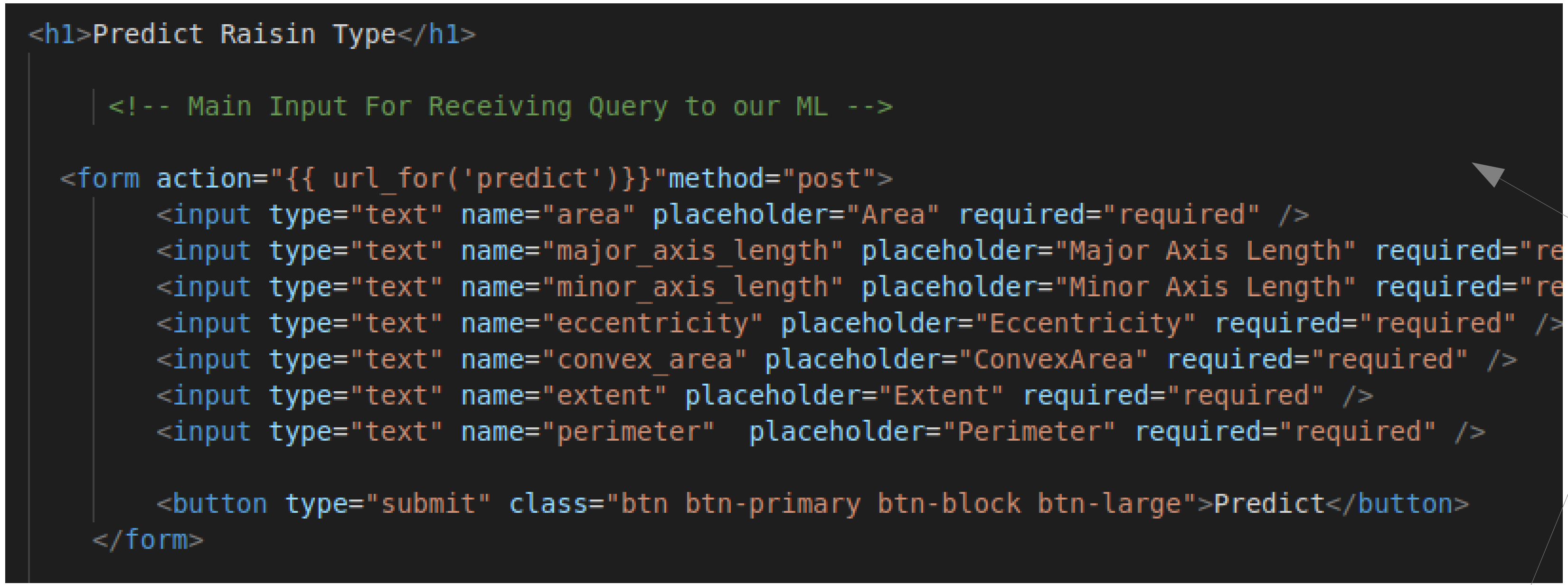
In the app index.html template is used. Since there are features in the model that differ from the features in demo app provided, it’s needed to modify the template. I changed the heading, added additional inputs, renamed them and also added a text about features and image. Also I changed some parameters in style.css file, e.g. changed background and text color.

Index.html is in templates, style.css is in static/css folder

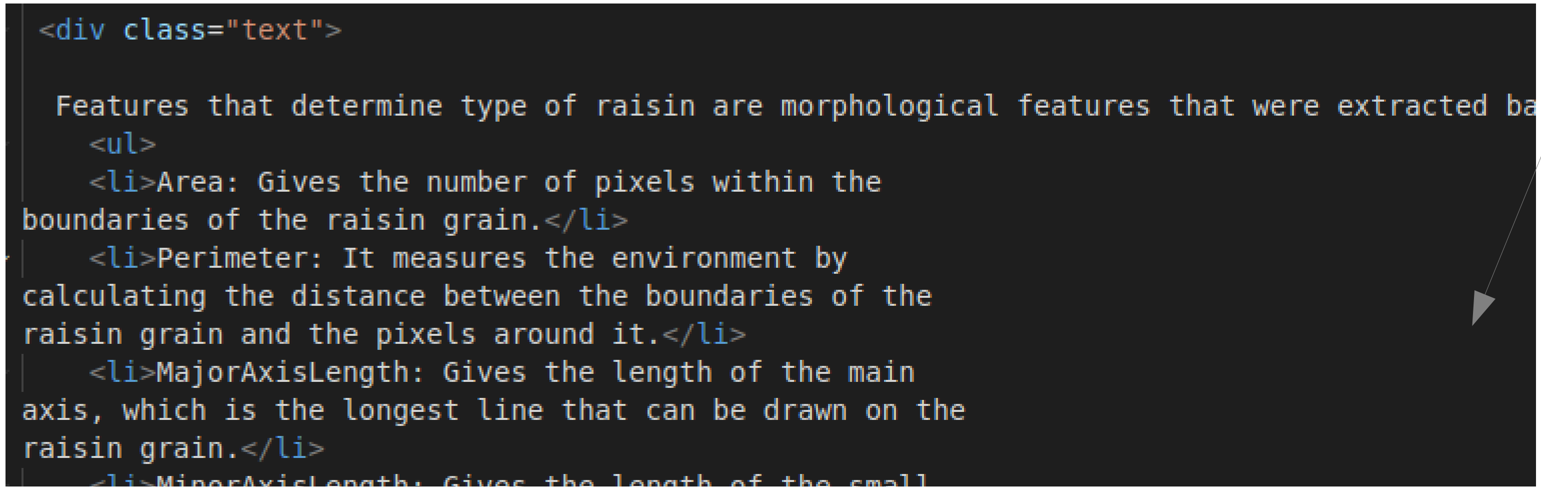
Made header ‘Predict Raisin Type’ and modified input fields.

Created a tag with class ‘text’ and inserted a short description.

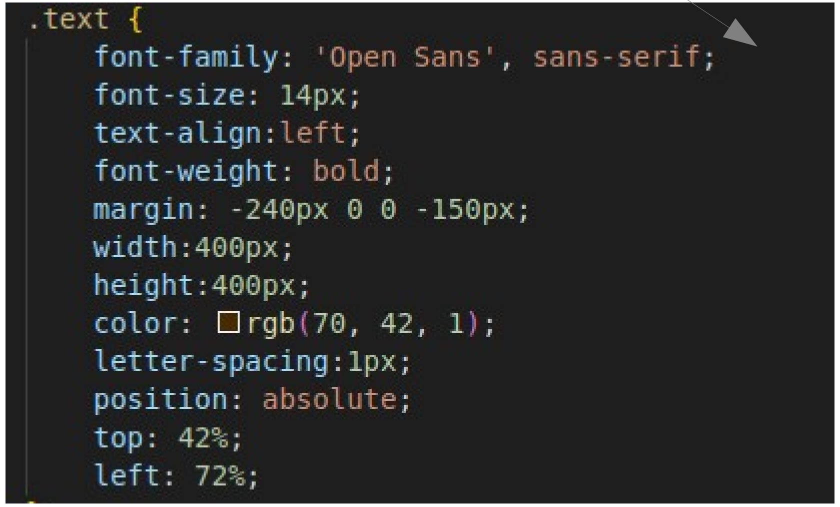
Added formatting for text tag in css file.



```html
<h1>Predict Raisin Type</h1>

    <!-- Main Input For Receiving Query to our ML -->

  <form action="{{ url_for('predict')}}"method="post">
        <input type="text" name="area" placeholder="Area" required="required" />
        <input type="text" name="major_axis_length" placeholder="Major Axis Length" required="re
        <input type="text" name="minor_axis_length" placeholder="Minor Axis Length" required="re
        <input type="text" name="eccentricity" placeholder="Eccentricity" required="required" />
        <input type="text" name="convex_area" placeholder="ConvexArea" required="required" />
        <input type="text" name="extent" placeholder="Extent" required="required" />
        <input type="text" name="perimeter"  placeholder="Perimeter" required="required" />

        <button type="submit" class="btn btn-primary btn-block btn-large">Predict</button>
    </form>
```



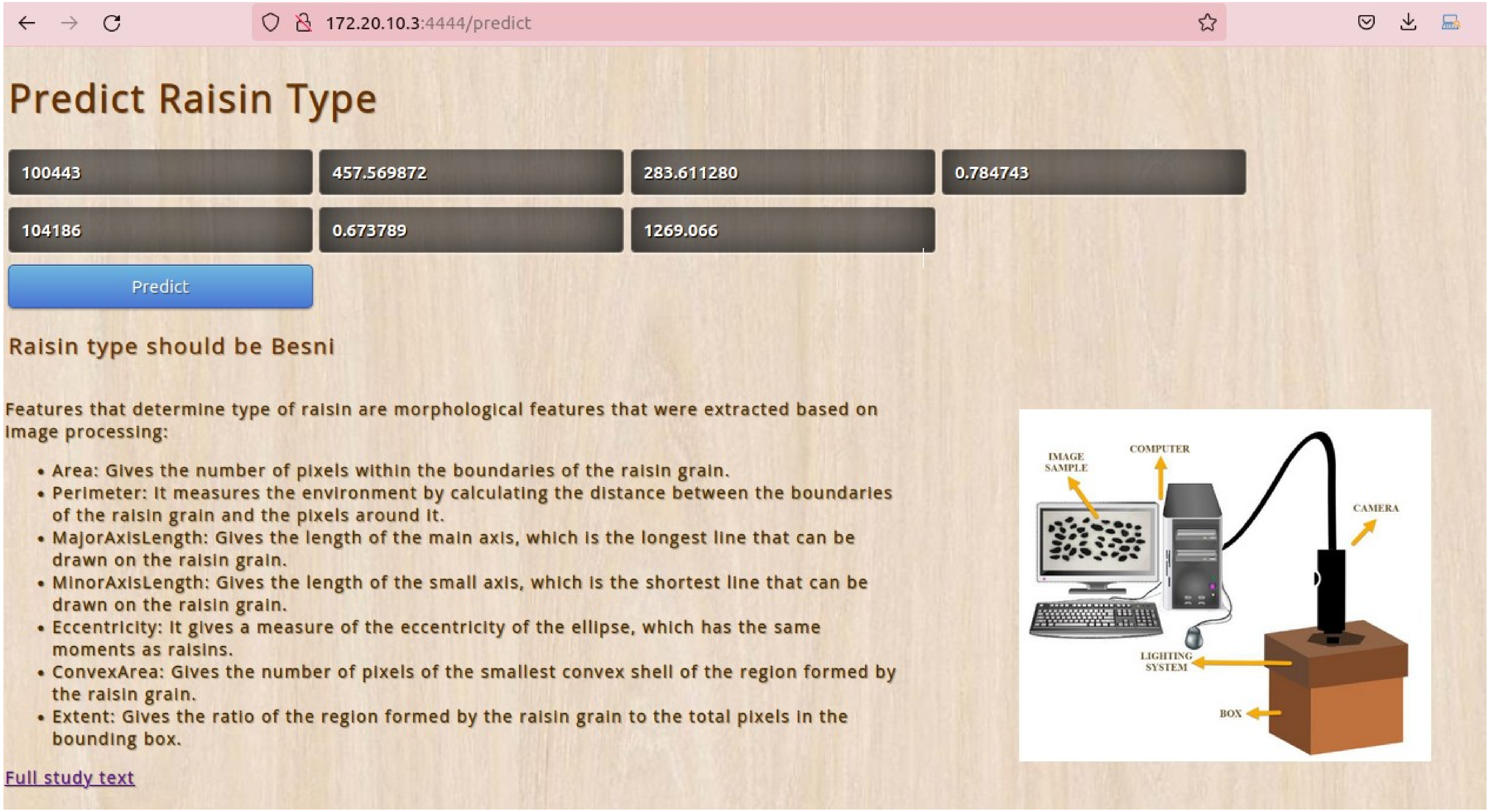
```html
 <div class="text">

  Features that determine type of raisin are morphological features that were extracted ba
    <ul>
    <li>Area: Gives the number of pixels within the
boundaries of the raisin grain.</li>
    <li>Perimeter: It measures the environment by
calculating the distance between the boundaries of the
raisin grain and the pixels around it.</li>
    <li>MajorAxisLength: Gives the length of the main
axis, which is the longest line that can be drawn on the
raisin grain.</li>
    <li>MinorAxisLength: Gives the length of the small
```



```css
.text {
     font-family: 'Open Sans', sans-serif;
     font-size: 14px;
     text-align:left;
     font-weight: bold;
     margin: -240px 0 0 -150px;
     width:400px;
     height:400px;
     color: ■rgb(70, 42, 1);
     letter-spacing:1px;
     position: absolute;
     top: 42%;
     left: 72%;
```

Testing the App

After committing the changes in the local repo I open console and type ‘python3 app.py’. The app is running on a local server with specified port. I test the app by typing features of single example from the test set. I can see the prediction and it’s correct.



172.20.10.3:4444/predict

Predict Raisin Type

100443	457.569872	283.611280	0.784743
104186	0.673789	1269.066	

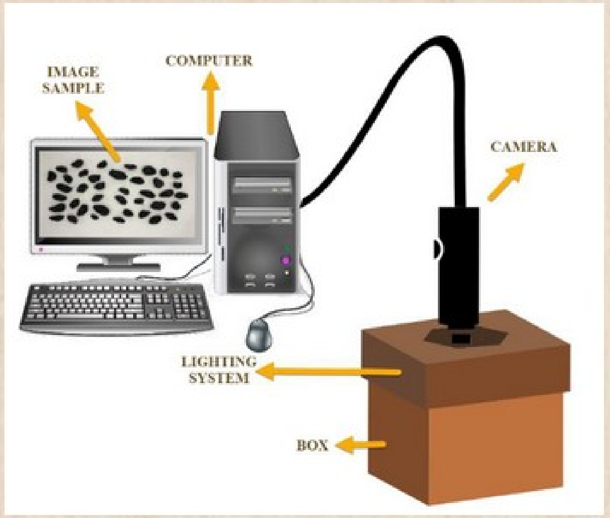
Predict

Raisin type should be Besni

Features that determine type of raisin are morphological features that were extracted based on image processing:

- Area: Gives the number of pixels within the boundaries of the raisin grain.
- Perimeter: It measures the environment by calculating the distance between the boundaries of the raisin grain and the pixels around it.
- MajorAxisLength: Gives the length of the main axis, which is the longest line that can be drawn on the raisin grain.
- MinorAxisLength: Gives the length of the small axis, which is the shortest line that can be drawn on the raisin grain.
- Eccentricity: It gives a measure of the eccentricity of the ellipse, which has the same moments as raisins.
- ConvexArea: Gives the number of pixels of the smallest convex shell of the region formed by the raisin grain.
- Extent: Gives the ratio of the region formed by the raisin grain to the total pixels in the bounding box.

Full study text



Deployment on Flask is completed.

Raisin type prediction app
Deployment on Flask

Data Glacier
Your Deep Learning Partner

Thank You