Data Mining and Adv. Statistical Modeling Mini-Project Report

Jeeva Mary Loui

Metro College Of Technology

Logistic Regression Using Python

Classification of Image

Abstract

Data collected from the properties of image sensed by a robot are used to classify the image to 7 unique classes using Logistic Regression model . The dataset is preprocessed for fitting it to Classification model . The performance is analysed and the model is made to achieve maximum efficiency.

Classify The image using Logistic Regression

Dataset

Image Segmentation Data Set The instances were drawn randomly from a database of 7 outdoor images. The images were hand-segmented to create a classification for every pixel. Each instance is a 3x3 region.

Data Description

- Number of instances: 2310
- Number of attributes: 19
- Number of Classes :7
- Target :Class

Attribute Information

1. region-centroid-col: the column of the center pixel of the region.
2. region-centroid-row: the row of the center pixel of the region.
3. region-pixel-count: the number of pixels in a region = 9.
4. short-line-density-5: the results of a line extraction algorithm that counts how many lines of length 5 (any orientation) with low contrast, less than or equal to 5, go through the region.
5. short-line-density-2: same as short-line-density-5 but counts lines of high contrast, greater than 5.
6. vedge-mean: measure the contrast of horizontally adjacent pixels in the region. There are 6, the mean and standard deviation are given. This attribute is used as a vertical edge detector.
7. vegde-sd: (see 6)

8. hedge-mean: measures the contrast of vertically adjacent pixels. Used for horizontal line detection.
9. hedge-sd: (see 8).
10. intensity-mean: the average over the region of (R + G + B)/3
11. rawred-mean: the average over the region of the R value.
12. rawblue-mean: the average over the region of the B value.
13. rawgreen-mean: the average over the region of the G value.
14. exred-mean: measure the excess red: (2R - (G + B))
15. exblue-mean: measure the excess blue: (2B - (G + R))
16. exgreen-mean: measure the excess green: (2G - (R + B))
17. value-mean: 3-d nonlinear transformation of RGB. (Algorithm can be found in Foley and VanDam, Fundamentals of Interactive Computer Graphics)
18. saturation-mean: (see 17)
19. hue-mean: (see 17)

Dataset source

Dataset comes from Datahub Machine Learning repository:

https://datahub.io/machine-learning/segment

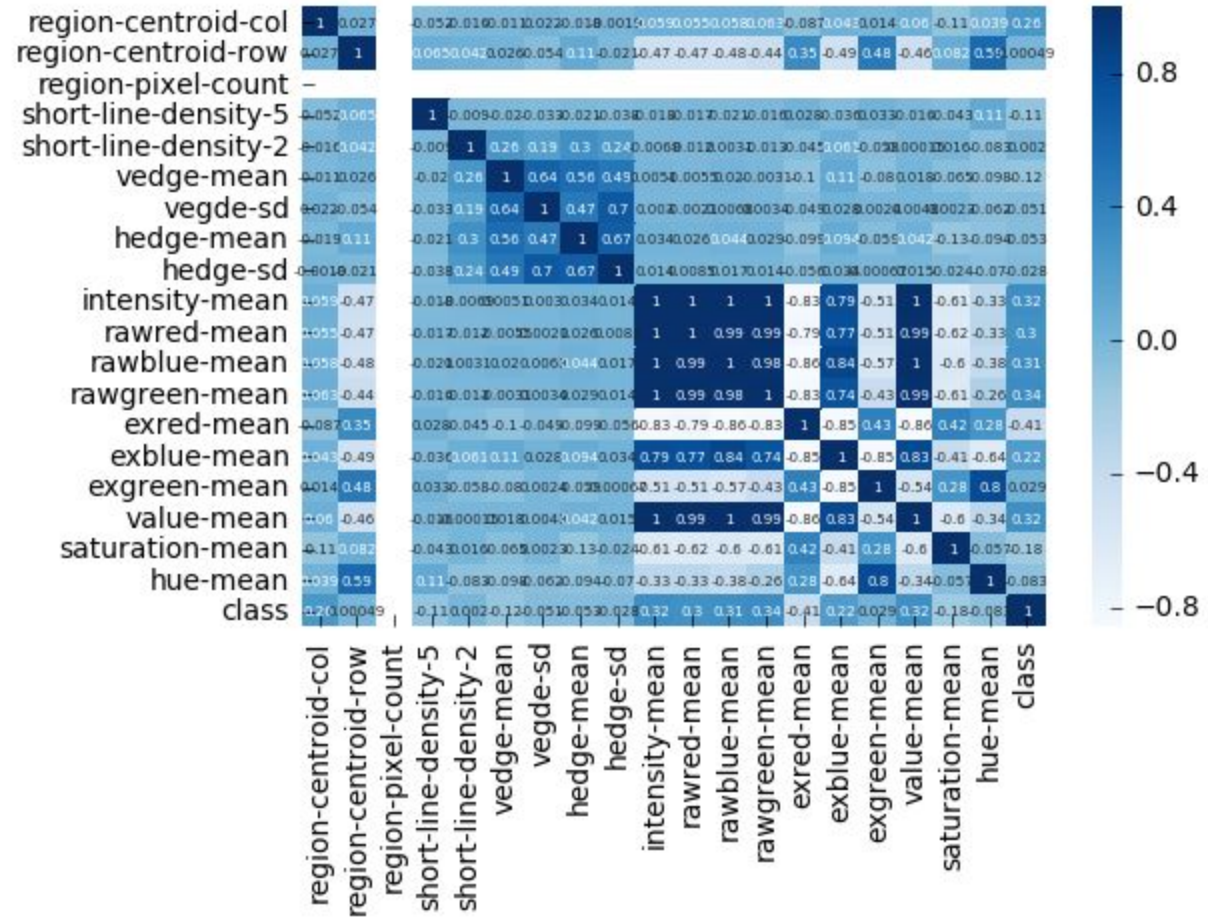
Data Preprocessing

In order to do Logistic regression for classification there has to be done some preprocessing.

Index	n-centro	centr	-pixel	line-der	-line-den	vedge-mean	vegde-sd	hedge-mean	hedge-sd	tensity-mea	awred-mea	awblue-mea	green-m	xred-mea	lue-me	green-me	ue-me	ration-n	lue-mea	class
0	218	178	9	0.11…	0	0.833333	0.547722	1.11111	0.544331	59.6296	52.4444	75.2222	51.2222	-21.55…	46.7…	-25.2…	75.…	0.31…	-2.04…	path
1	113	130	9	0	0	0.277778	0.250924	0.333333	0.365148	0.888889	0	2.55556	0.1111…	-2.666…	5	-2.33…	2.5…	1	-2.12…	foliage
2	202	41	9	0	0	0.944448	0.772202	1.11111	1.0256	123.037	111.889	139.778	117.444	-33.44…	50.2…	-16.7…	139…	0.19…	-2.29…	sky
3	32	173	9	0	0	1.72222	1.78159	9	6.74949	43.5926	39.5556	52.8889	38.3333	-12.11…	27.8…	-15.7…	52.…	0.26…	-1.99…	path
4	61	197	9	0	0	1.44444	1.51535	2.61111	1.92546	49.5926	44.2222	61.5556	43	-16.11…	35.8…	-19.7…	61.…	0.30…	-2.02…	path

```
multi=pd.read_csv('segment_csv.csv')
#Preprocessing
desc=multi.describe()
multi.dtypes
#Encoding
c = multi['class'].astype('category')
d = dict(enumerate(c.cat.categories))
print (d)
multi['class'] = multi['class'].astype('category').cat.codes
#multi['back'] = multi['class'].map(d)
multi.dtypes
multi.isnull().sum()
multi['class'].value_counts()
#Correlation Analysis
multi.corr()
sns.heatmap(multi.corr(),cmap='Blues' ,annot= True,annot_kws={"size": 5} )
##Extract independant and response variables
target=multi['class']
X=multi.drop(['class'], axis=1)
y=target
```

There are no missing values and the only non numerical attribute is Class.The Class has 7 different categories and It is encoded to numerical using cat.codes function in pandas as{0: 'brickface', 1: 'cement', 2: 'foliage', 3: 'grass', 4: 'path', 5: 'sky', 6: 'window'}.A correlation analysis is done to understand the dependant and target attributes. The target is extracted from the dataset and classification is based on the other 19 attributes.



Now it's important to check if any attribute can be eliminated to do the regression so the model is much efficient, the technique I chose to use is Recursive Feature Elimination.

```
#Recursive Feature Elimination
from sklearn.linear_model import LogisticRegression
from sklearn.feature_selection import RFE
accuracies = []
feature_set = []
max_accuracy_so_far = 0
for i in range(1,len(X.columns)+1):
    selector = RFE(LogisticRegression(), i,verbose=1)
    selector = selector.fit(X, y)
    current_accuracy = selector.score(X,y)
    accuracies.append(current_accuracy)
    feature_set.append(selector.support_)
    if max_accuracy_so_far < current_accuracy:
        max_accuracy_so_far = current_accuracy
        selected_features = selector.support_
    print('End of iteration no. {}'.format(i))
X_sub = X.loc[:,selected_features]
```

The selected feature is boolean list , Which has no False value and all the 19 attributes are important to classify the image according to Recursive Feature Elimination

Train and Build a Logistic Regression Model

The dataset has to be split into the train and test to train a regression model and test the model using the test set. The cross_validation in scikit package helps doing this efficiently. The training set is fit to a logistic regression model.

```
#Spliting of dataset
from sklearn.cross_validation  import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X_sub,y,random_state=0)
#train the model
from sklearn.linear_model import LogisticRegression
model = LogisticRegression()
#train the model
model.fit(X_train,y_train) #training
```

Performance Analysis

Now the model is ready and model performance has to evaluated .

```
#predict testcases
y_pred = model.predict(X_test)
y_pred_probs = model.predict_proba(X_test)
#performance measures on the test set
from sklearn.metrics import classification_report, confusion_matrix,accuracy_score
accuracy_score(y_test,y_pred)
confusion_matrix(y_test,y_pred)
print(classification_report(y_test,y_pred))
```

accuracy_score(y_test,y_pred): 0.93252595155709339

Which is really good accuracy score and the classification can be said 93.2% accurate.

Confusion Matrix:

	0	1	2	3	4	5	6
0	85	0	1	0	0	0	2
1	0	61	0	0	0	0	9
2	0	0	64	0	1	0	8
3	0	0	0	89	0	0	0
4	0	0	0	0	84	0	0
5	0	0	0	0	0	89	0
6	0	5	13	0	0	0	67

The confusion matrix shows the classification is excellent and most of the classification fall in True positives.

Classification Report :

	precision	recall	f1-score	support
0	1.00	0.97	0.98	88
1	0.92	0.87	0.90	70
2	0.82	0.88	0.85	73
3	1.00	1.00	1.00	89
4	0.99	1.00	0.99	84
5	1.00	1.00	1.00	89
6	0.78	0.79	0.78	85
avg / total	0.93	0.93	0.93	578

The classification report has an average precision recall and f1 score of 93 %, which is amazing. And the model has achieved a good perfection .

Conclusion

The Image Segmentation Data Set has been pre processed efficiently and Classification Model is fit with the data and the image can be predicted accurately using this model. The performance scores tells that model is amazing and has achieved good level of perfection with supporting performance scores.