PMDS505P Data Mining and Machine Learning
Experiment 6

February 2025

1 Work to do today

Note: Make a single pdf file of the work you are doing in jupyter notebook. Upload with proper format. Please mention your name and roll no properly with Experiment number in the first page of your submission.

Performance measures: Cross validation

Q1. Today we will try to perform cross validation to check how well the model generalizes to a unseen data. We will see how to implement K fold cross validation and stratified K fold cross validation.

- Download the liver patient dataset that you have with you.
- Perform Min-Max scaling
- Do the train test split of the data with test size 20%.
- Fit the LogisticRegression model to the this training data.
- Print the accuracy which is also a performance measure as far as a classification problem is concerned with.
- Now import Kfold and cross_val_score functions from the available modules for performing Kfold cross validation.

  from sklearn.model_selection import KFold, cross_val_score
- Now you can create a new object of the class LogisticRegression as logisticR

  logisticR = LogisticRegression()
- now let us perform the KFold class to split the data into 5 folds. you can use the code.

  kfold_validation = KFold(n_splits=5, shuffle=True,random_state=42)

- Now we will use cross_val_score function to perform the 5 fold cross validation and print the accuracy scores in each case.

  results = cross_val_score(logisticR, _train,y_train,scoring='accuracy', cv=kfold_validation)

- Now the cross-validation scores can be printed.

  print("Cross-validation accuracy scores:", results)
  print("Mean Accuracy:", np.mean(results))

- Now we can try how we can implement StratifiedKFold cross validation.

  from sklearn.model_selection import StratifiedKFold

- as we did the splitting in the above case where we had Kfold, we can try the same in this case. Create an obeject of the stratifedKFold class and use it as KFold to do the splitting. Further use the Cross_val_score to print the cross validation scores as we did in KFold cross validation.

  skf = StratifiedKFold(n_splits=5, shuffle=True, random_state=42)

  scores = cross_val_score(logisticR,X_train,y_train, scoring='accuracy',cv=skf)
  print("Stratified CV Scores:", scores)
  print("Mean accuracy:", scores.mean())

Performance measures: Confusion matrix, Precision, Recall, F1 score

- Q2. Now fit the logistic regression model for the liver patient data without performing cross validation with a train test split of 80:20.
- Print the accuracy score.
- we can also print the confusion matrix, precision and recall. The same was discussed in the class.
- import all these measures to be evaluated from appropriate modules

  from sklearn.metrics import confusion_matrix, precision_score, recall_score, f1_score, ConfusionMatrixDisplay.

  cm = confusion_matrix(y_test, y_pred)
  print("Confusion Matrix:", cm)

  precision = precision_score(y_test, y_pred)
  recall = recall_score(y_test, y_pred)

```
f1 = f1_score(y_test, y_pred)

print("Precision:", precision)
print("Recall:", recall)
print("F1 Score:", f1)
```

- The same confusion matrix if you want can be printed in a more better manner using the below code.

```
# Plot confusion matrix using seaborn heatmap import seaborn as sns
plt.figure(figsize=(6, 4))
sns.heatmap(cm, annot=True, fmt="d", cmap="Blues", xticklabels=["No Disease",
"Disease"], yticklabels=["No Disease", "Disease"])
plt.xlabel("Predicted Label")
plt.ylabel("True Label")
plt.title("Confusion Matrix")
plt.show()
```

Q3. Explore what is an ROC curve and AUC. And plot the same in the above problem. Mention what they specify in the above case.

Deicsion trees: Regression and classification

- Let's try to fit a decision tree for a classification problem and view the same and see how the predictions can be made using the same.
- We will use the same liver disease dataset to fit the decision tree model.
- Prepare the data by doing the necessary scaling and train test split.
- Now let us import the necessary class to perform the same.

```
from sklearn.metrics import accuracy_score
from sklearn.tree import DecisionTreeClassifier
```

- Fit the decision tree model.

```
model = DecisionTreeClassifier()
model.fit(X_train, y_train)
```

- Let us see the predictions made by the model.

```
predictions1 = model.predict(X_test)
print(accuracy_score(y_test,model.predict(X_test)))
print(predictions1)
```

- We can print and check the Decision tree.

  from sklearn.tree import plot_tree plt.figure(figsize=(100,50), dpi=150)
  plot_tree(model)

- Zoom in the image and visualize the outputs. This fitting we are performing until the Gini index of each region is 0. So the leaf nodes are pure nodes.

- Compare the accuracy of logistic regression model and decision tree model you have fitted.

- Now use the Book1.csv file we used in multiple regression fitting in Labsheet 3.

- Do the necessary preprocessing of the data and train test split of the data and fit a multiple regression model and compute the error.

- Now use the DecisionTreeREgressor class to fit a decision tree model and check the decision tree and errors.

  from sklearn.tree import DecisionTreeRegressor
  from sklearn.metrics import mean_squared_error
  model = DecisionTreeRegressor()
  model.fit(X_train, y_train)
  predictions1 = model.predict(X_test)
  print(mean_squared_error(y_test,model.predict(X_test)))

- Now plot the decision tree as done in the classifier case above.

- Further also compare the errors you received in both the case and print and compare them.

- Now this what you see is a basic decision tree model. But there are many improved methods which we will implement in the coming classes which can give better accuracy than these.