Results and Documentation

Results of Regression and Classification Models

Regression Models

Two regression models were evaluated on the test data, and here are the results:

Mean Square Error Comparison:

LR Model 1 (Linear Regression): MSE = 0.0742
LR with SGD (Linear Regression with Stochastic Gradient Descent): MSE = 0.0845

The lower the MSE, the better the model's predictive performance. In this case, LR Model 1 has a lower MSE, indicating that it makes more accurate predictions on average compared to LR with SGD. Thus, LR Model 1 is better in terms of MSE.

R-squared (R^2) Score Comparison:

LR Model 1 (Linear Regression): R^2 Score = 0.3958
LR with SGD (Linear Regression with Stochastic Gradient Descent): R^2 Score = 0.3071

The R^2 score measures how well the model explains the variance in the data. A higher R^2 score indicates a better fit. LR Model 1 has a higher R^2 score, indicating that it explains more of the variance in the restaurant ratings compared to LR with SGD. Thus, LR Model 1 is better in terms of R^2 score.

Classification Model

Performance Evaluation of Logistic Regression Model:

Accuracy: The model achieved an accuracy score of approximately 0.9205, which is quite good. It means that about 92% of the records were classified correctly.

Observation:

The model is relatively good at classifying records as Class 1 ('Poor' and 'Average'). It has a high True Positive rate (1569 TP) for Class 1.
The model also performs reasonably well in Class 2 ('Good', 'Very Good', and 'Excellent'), with a high True Negative rate (364 TN).
There are some false positives (34 FP) and false negatives (133 FN), indicating that the model makes some errors in classifying records.

The choice of features and the logistic regression model seem to be effective in distinguishing between 'Poor'/'Average' and 'Good'/'Very Good'/'Excellent' records. Overall, the model is quite good at distinguishing between the two classes, but there is room for improvement, especially in reducing false positives and false negatives. Performance Report for Confusion Matrices:

Random Forest Classifier and Decision Tree Classifier: Both achieved perfect accuracy on the test data, with no misclassifications (0 false positives and 0 false negatives). This suggests that these models fit the training data very well, possibly overfitting it.

Logistic Regression: It also performed quite well with an accuracy score of 0.9205. It had a moderate number of false positives and false negatives compared to the perfect scores of the tree-based models.

K-Neighbors Classifier: It had the lowest accuracy score of 0.8314 and a higher number of false positives and false negatives compared to the other models, indicating that it might not fit the data as well as the tree-based models.

GitHub Commands Used

Create a Git Repository in GitHub:
- Initialized a local Git repository: `git init`
- Staged all the files in the current directory for commit: `git add .`
- Committed the staged files: `git commit -m "Initial commit"`

Connect Local Repository to GitHub:
- Added a remote repository named 'origin' on GitHub: `git remote add origin https://github.com/diwasp1/predictive_modelling.git`
- Verified the remote repository has been added correctly: `git remote -v`

Push the Code to GitHub:
- Pushed the code to the 'master' branch in GitHub: `git push -u origin master`
- Changed the default branch to 'master' and deleted the 'main' branch on GitHub: (Performed in the GitHub interface)
- To see all the branches in local and on GitHub: `git branch -a`

Docker Image Creation and Push
- Create a Docker file: ` touch dockerfile`
- Add Instruction in dockerfile
- Build a Docker image: `docker build . -t diwasp1/predictivemodelling `
- Push the Docker image to Docker Hub: `docker push diwasp1/ predictivemodelling`
- Run the Docker container: `docker run -p 8888:8888 diwasp1/ predictivemodelling`

Links
- Link to the Tableau Dashboard
- Source code on GitHub
- Docker Image on Docker Hub