Assume that you are hired by LinkedIn to investigate issues related to the continued use of their platform. Work with the “LinkedIn Continued Use” data mentioned in the slides. You need to conduct the following tests in Python to answer relevant questions.

Part 1: Gradient Boosting - Binary

* Run XGBoost (target variable *stay)* following the instructions on the slides.
* Generate the feature importance and model performance measures.
* You should be able to replicate the results on the slides.
* Interpret the outputs.

The model performance measures for the XGBoost binary classification model are:

* Error rate: 0.155
* Sensitivity: 0.839
* Specificity: 0.850
* Precision: 0.812
* AUC: 0.874
* Cross-Entropy: 0.775

These measures indicate that the model is performing reasonably well on the test set.

The error rate is relatively low, which means that the model is making correct predictions for the majority of the cases.

the sensitivity and specificity values are also high, indicating that the model is able to correctly identify positive and negative cases.

the precision value is also high, indicating that the model is able to correctly identify a high proportion of true positive cases among all positive predictions.

The AUC value is also high, indicating that the model has good discriminative power between positive and negative cases.

The cross-entropy value is relatively low, indicating that the model is able to accurately estimate the probability of each class for each instance in the test set.

The feature importance for the XGBoost binary classification model are:

* feature 1: 0.292
* feature 2: 0.105
* feature 3: 0.080
* feature 4: 0.129
* feature 5: 0.053
* feature 6: 0.077
* feature 7: 0.204
* feature 8: 0.060

These values represent the relative importance of each feature in the model. The higher the value, the more important the feature is in predicting the target variable

* AdaBoost vs. XGBoost
  + Run AdaBoost using the codes on the slides for AdaBoost
  + Compare the results

|  |  |
| --- | --- |
| Ada Boost | XG boost |
| Error rate: 0.282 | Error rate: 0.155 |
| Sensitivity: 0.710 | Sensitivity: 0.839 |
| Specificity: 0.725 | Specificity: 0.850 |
| Precision: 0.667 | Precision: 0.812 |
| AUC: 0.717 | AUC: 0.874 |
| Cross-Entropy: 10.153 | Cross-Entropy: 0.775 |

it seems that the XGBoost model outperformed the AdaBoost model on all metrics except for

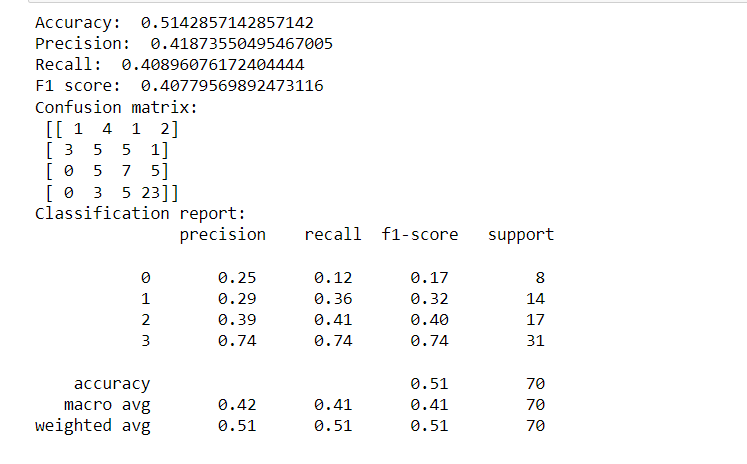
precision

The XGBoost model had a lower **error rate** (0.155 vs 0.282), higher **sensitivity** (0.839 vs 0.710), higher **specificity** (0.850 vs 0.725), higher **AUC** (0.874 vs 0.717), and much lower **cross-entropy** (0.775 vs 10.153) compared to the AdaBoost model. However, the AdaBoost model had a slightly higher **precision** (0.667 vs 0.812) compared to the XGBoost model.

Part 2: Gradient Boosting – Multi-class

* Run XG Boast (target variable *Q5\_1)* following the instructions on lecture slides.
* Generate the feature importance and model performance measures.
* You should be able to replicate the results on the slides.

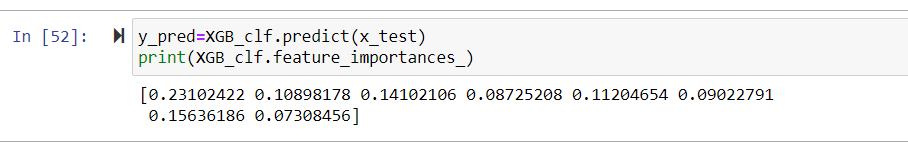
performance measures:



The output shows the results of evaluating a multi-class XGBoost model:

* The accuracy of the model is 0.51, indicating that 51% of the instances were classified correctly.
* The precision, recall, and F1 score are 0.42, 0.41, and 0.41, respectively. These metrics give a measure of how well the model performs in terms of identifying instances for each class, with a higher value indicating better performance.
* The confusion matrix provides a summary of the number of correct and incorrect predictions for each class. For example, the model correctly predicted 23 instances in class 3.
* The classification report provides a detailed breakdown of the precision, recall, and F1 score for each class, as well as the support, which is the number of instances in each class.

feature importance:

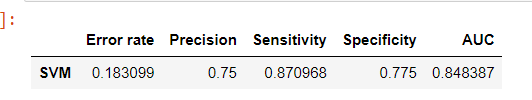


These values represent the relative importance of each feature in the model. The higher the value,

the more important the feature is in predicting the target variable.

Part 3: SVM - Binary

* Run XG Boast (target variable *stay)* following the instructions on lecture slides.
* Generate the feature importance and model performance measures.
* You should be able to replicate the results on the slides.
* Interpret the outputs.

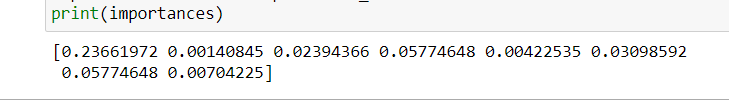


The table shows five performance metrics of the model: error rate, precision, sensitivity, specificity,

and AUC.

* The error rate is the proportion of incorrect predictions made by the model. the model makes incorrect predictions about 18.31% of the instances.
* The precision is the proportion of correct positive predictions out of all positive predictions made by the model. In this case, the model correctly identifies 75% of the positive instances.
* The sensitivity is the proportion of correct positive predictions out of all actual positive instances in the dataset. The model correctly identifies 87.10% of the positive instances.
* The specificity is the proportion of correct negative predictions out of all actual negative instances in the dataset. The model correctly identifies 77.50% of the negative instances.
* The AUC is a measure of the overall performance of the model in distinguishing between positive and negative instances. AUC ranges between 0 and 1, with higher values indicating better performance. The model has an AUC of 0.848387 or 84.84%, which means that it performs well in distinguishing between positive and negative instances.

Feature importance:



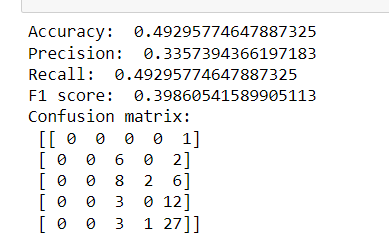
These values represent the relative importance of each feature in the model. The higher the value,

the more important the feature is in predicting the target variable.

Part 4: SVM – Multi-class

* Run XG Boast (target variable *Q5\_1)* following the instructions on the slides
* Generate the feature importance and model performance measures
* You should be able to replicate the results on the slides.

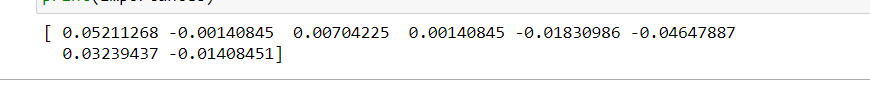
performance measures:



The output suggests that the model is a multi-class classification model.

* The model has an accuracy of 0.49295774647887325, which means that it correctly classified 49.3% of the total instances in the test dataset.
* The model has a precision of 0.3357394366197183, which means that among all the instances predicted as positive, only 33.6% are actually positive.
* The model has a recall of 0.49295774647887325, which means that the model correctly identified 49.3% of all positive instances.
* The model has an F1 score of 0.39860541589905113, which is the harmonic mean of precision and recall. It is a measure of the balance between precision and recall.
* The model has an error rate of 0.5070422535211268, which is the complement of accuracy. It means that the model misclassified 50.7% of the instances.
* The confusion matrix shows the number of correct and incorrect predictions for each class

feature importance:



These values represent the relative importance of each feature in the model. The higher the value,

the more important the feature is in predicting the target variable.

Part 5: Gradient Boosting vs. SVM

* Run the following:
  + Target variable *stay*
  + IVs: Q7\_1 to Q7\_19
* Generate the model performance measures and compare the results
* Interpret the outputs

1)performance measures

|  |  |
| --- | --- |
| SVM | Gradient Boosting |
| Error rate: 0.2571428571428571 | Error rate: 0.214 |
| Precision: 0.7 | Precision: 0.727 |
| Sensitivity: 0.7 | Sensitivity: 0.800 |
| Specificity: 0.775 | Specificity: 0.775 |
| AUC: 0.744 | AUC: 0.858 |

it seems that the Gradient Boosting model outperformed the SVM model on all metrics.

* The SVM model has an error rate of 0.257, while the Gradient Boosting model has an error rate of 0.214. This means that the Gradient Boosting model has a lower error rate
* The SVM model has a precision of 0.7, while the Gradient Boosting model has a precision of 0.727. This means that the Gradient Boosting model is better at correctly predicting positive instances.
* The SVM and Gradient Boosting models have sensitivities of 0.7 and 0.8 respectively. This means that the Gradient Boosting model is better at correctly identifying positive instances in the test data.
* Both models have a specificity of 0.775, meaning they are equally good at correctly identifying negative instances.
* The SVM model has an AUC of 0.744, while the Gradient Boosting model has an AUC of 0.858. This means that the Gradient Boosting model has a better trade-off between sensitivity and specificity, making it better at predicting both positive and negative instances.