David Galati University of Twente June 2024

AAIB Assignment 5

Acknowledgement: ChatGPT was used both in the generation of text and code

Task 1

Following what criteria business can perform model selection? Review notebooks 1-5, and ponder about how and why would you suggest a model for deployment in each practical/case (the cases with multiple models used). (Min 250 words, Max 1 A4 page)

Notebook 1

In this notebook, we used Logistic Regression and Random Forest to predict whether users clicked on ads. The key criteria for model selection in this context include accuracy, interpretability, and processing time. Logistic Regression, with its high interpretability, is useful for understanding which features influence ad clicks. Random Forest, on the other hand, provides higher accuracy and handles non-linear relationships well. For deployment, Random Forest would be suggested due to its superior performance and robustness, while Logistic Regression can be used alongside for insights into feature importance.

Notebook 2

This notebook focused on customer churn prediction using XGBoost. The criteria here include handling imbalanced data, predictive power, and scalability. XGBoost was chosen for its ability to manage imbalanced datasets effectively and its high accuracy. However, simpler models like Logistic Regression or Decision Trees could be considered for initial deployment to ensure ease of interpretation and faster implementation. As the business grows and requires more precise predictions, transitioning to XGBoost would be advantageous due to its superior performance.

Notebook 3

In this notebook, k-Means was used to segment customers. The primary criteria for model selection in clustering tasks include the ability to handle the size and distribution of data, interpretability, and implementation simplicity. k-Means, with its ease of implementation and interpretation, is ideal for initial segmentation tasks. However, for more complex and non-globular data distributions, models like DBSCAN or Hierarchical Clustering would be recommended. These models handle noise and varying cluster sizes better, providing more meaningful segments for targeted marketing.

Notebook 4

Here, we implemented Logistic Regression, SVM, Decision Trees, Random Forest, AdaBoost, and Gradient Boosting to predict hotel booking cancellations. The criteria include accuracy, handling of complex patterns, and computational efficiency. Ensemble models like Random Forest and Gradient Boosting often outperform individual models due to their ability to capture complex patterns and reduce overfitting. For deployment, Random Forest would be a strong choice due to its balance of performance and efficiency. Gradient Boosting can also be considered for its high accuracy, especially if computational resources allow.

Notebook 5

This notebook used various models, including Logistic Regression, SVM, Decision Trees, and more, for predicting hotel bookings. The selection criteria involve accuracy, interpretability, and model complexity. Logistic Regression is valuable for its simplicity and interpretability, making it a good choice for initial deployment and quick insights. For improved accuracy and handling non-linear relationships, Random Forest or Gradient Boosting would be recommended. These models provide a better balance between performance and computational requirements, making them suitable for scaling.

Task 2

Explore the practical5 jupyter notebook, fix 10 existing errors (e.g., semantic, logical, runtime, etc.), and submit notebook without errors.

Reflect on the notebook structure: Steps on data preprocessing, model implementation. What what three top things you would do differently (better)? List what would you do differently and explain why.

Table 1 Mistakes and their respective fixes

Mistake	Fix
`From metrics import confusion_matrix`	`from sklearn.metrics import confusion_matrix`
`data = pd.read_csv('hotel_bookings.csv')`	`data = pd.read_csv('datasets/hotel_bookings.csv')`
`X = data.iloc[]`	`X = data.iloc[:, 1:]`
`X_train, X_test, y_train y_test = train_test_split(X, y, test_size = 0.2, random_state = 0)`	`X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state = 0)`
`X_train = scc.fit_transform(X_train)`	`X_train = sc.fit_transform(X_train)`
`ct = make_column_transformer( (OneHotEncoder(),['meal''distribution_channel','reservation_status','country','arrival_date_month','market_segment','deposit_type','customer_type', 'reserved_room_type','assigned_room_type']), remainder = 'passthrough')`	Missing coma: `'meal', 'distribution_channel'` This one was so hard to find :))
`ac = accuracy_score(y_train, y_pred)`	`ac = accuracy_score(y_test, y_pred)`
`clf_ab = AdaBoostClassifier(n_estimators=100, learning_rate=10000000, random_state=0)`	`clf_ab = AdaBoostClassifier(n_estimators=100, learning_rate=1, random_state=0)` learning rate really high
`pca = PCA(n_components = 120)`	`pca = PCA(n_components = 50)` to match the text

`classifier.fit(X_train, y_train)`	`classifier.fit(X_train, x_train)`
`ac = accuracy_score(y_test, y_test)`	`ac = accuracy_score(y_test, y_pred)`

The fully working notebook is enclosed in the submission files.

Table 2 Personal improvements

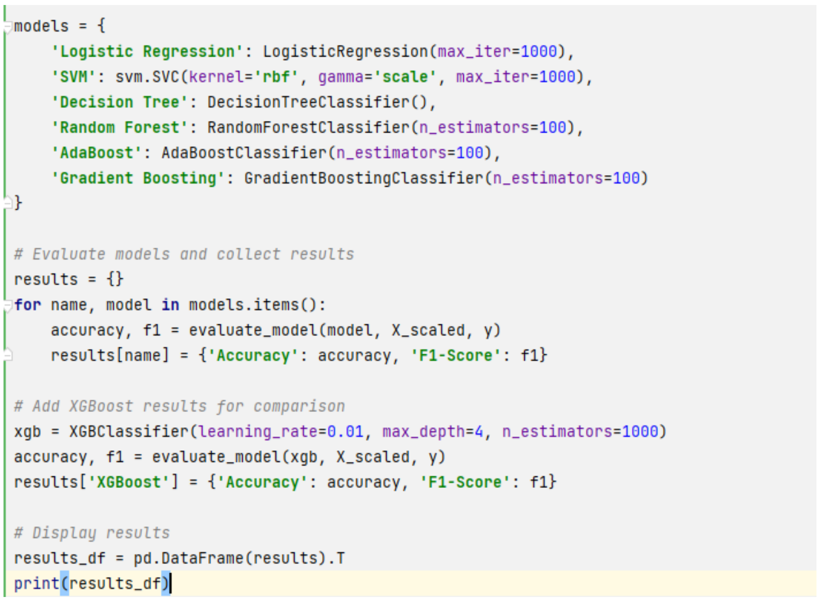
Aspect	Current Approach	Improvement	Reason
Feature Engineering	OneHotEncoder for all categorical variables	Use Target Encoding or Frequency Encoding for high cardinality features	Reduce dimensionality and computational cost, retain information
Handling Imbalanced Data	Not explicitly addressed	Apply Synthetic Minority Oversampling Technique (SMOTE) or similar techniques before scaling	Ensure balanced dataset for training, improve generalization
Hyperparameter Tuning	Not systematically implemented	Use GridSearchCV or RandomizedSearchCV with cross-validation	Optimize model performance, ensure robust validation

Task 3

In notebook 2 from practical 2 (Lecture 2), we did implement many DPT tasks and simply implemented Xboosting Classifier. In this task, you are asked to implement at least 5 other classifiers, use cross-validation 5-Fold and extract metrics of validation (at least 2), and compare/discuss results.

Logistic Regression, SVM, Decision Tree, Random Forest, AdaBoost and Gradient Boosting Models were used.

Table 3 discusses the results of each model based on the accuracy and F1-score:



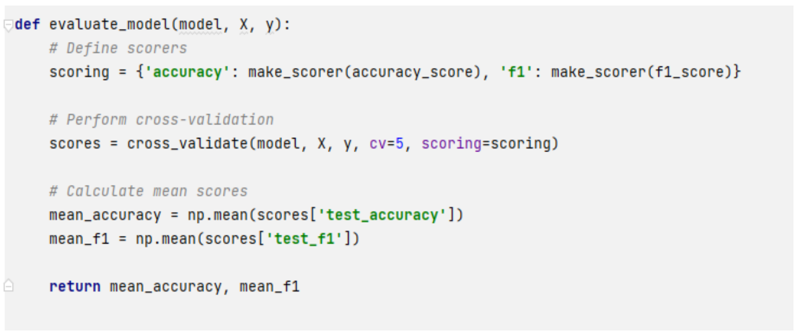
```python
models = {
    'Logistic Regression': LogisticRegression(max_iter=1000),
    'SVM': svm.SVC(kernel='rbf', gamma='scale', max_iter=1000),
    'Decision Tree': DecisionTreeClassifier(),
    'Random Forest': RandomForestClassifier(n_estimators=100),
    'AdaBoost': AdaBoostClassifier(n_estimators=100),
    'Gradient Boosting': GradientBoostingClassifier(n_estimators=100)
}

# Evaluate models and collect results
results = {}
for name, model in models.items():
    accuracy, f1 = evaluate_model(model, X_scaled, y)
    results[name] = {'Accuracy': accuracy, 'F1-Score': f1}

# Add XGBoost results for comparison
xgb = XGBClassifier(learning_rate=0.01, max_depth=4, n_estimators=1000)
accuracy, f1 = evaluate_model(xgb, X_scaled, y)
results['XGBoost'] = {'Accuracy': accuracy, 'F1-Score': f1}

# Display results
results_df = pd.DataFrame(results).T
print(results_df)
```

Figure 1 The 6 implemented models



```python
def evaluate_model(model, X, y):
    # Define scorers
    scoring = {'accuracy': make_scorer(accuracy_score), 'f1': make_scorer(f1_score)}

    # Perform cross-validation
    scores = cross_validate(model, X, y, cv=5, scoring=scoring)

    # Calculate mean scores
    mean_accuracy = np.mean(scores['test_accuracy'])
    mean_f1 = np.mean(scores['test_f1'])

    return mean_accuracy, mean_f1
```

Figure 2 Accuracy and F1 score using cross-validation.

Table 3 Discussion and Comparison of the used models

Model	Accuracy	F1-Score	Discussion
Logistic Regression	0.808179	0.616100	Strong performance: The logistic regression model shows a high accuracy and a good F1-score, indicating a reliable performance in distinguishing between classes. This makes it a robust baseline model for comparison.
SVM	0.776377	0.587013	Moderate performance: SVM performs slightly lower in accuracy and F1-score compared to logistic regression. It indicates that while SVM is effective for certain data distributions, it might not be the best fit for this dataset.
Decision Tree	0.730655	0.496140	Low performance: The decision tree model has the lowest accuracy and F1-score among the models, likely due to overfitting on the training data. This model may not generalize well to unseen data.
Random Forest	0.793557	0.561588	Moderate performance: Random Forest improves on the decision tree, but still lags behind logistic regression. This model benefits from ensemble learning, reducing overfitting seen in single decision trees.
AdaBoost	0.805764	0.612348	Good performance: AdaBoost shows comparable accuracy to logistic regression but slightly lower F1-score. It indicates that boosting weak classifiers enhances performance significantly.

Gradient Boosting	0.806191	0.597905	Good performance: Gradient boosting also performs well, with accuracy and F1-score close to those of AdaBoost and logistic regression. It is effective in handling this dataset.
XGBoost	0.807895	0.602455	Good performance: XGBoost has high accuracy and a good F1-score, similar to logistic regression. Known for its efficiency, it is often a top choice in machine learning competitions and practical applications.