



Model Optimization and Tuning Phase Template

Date	June 2024
Team ID	739865
Project Title	Prosperity Prognosticator: Machine Learning for startup success prediction
Maximum Marks	10 Marks

Model Optimization and Tuning Phase

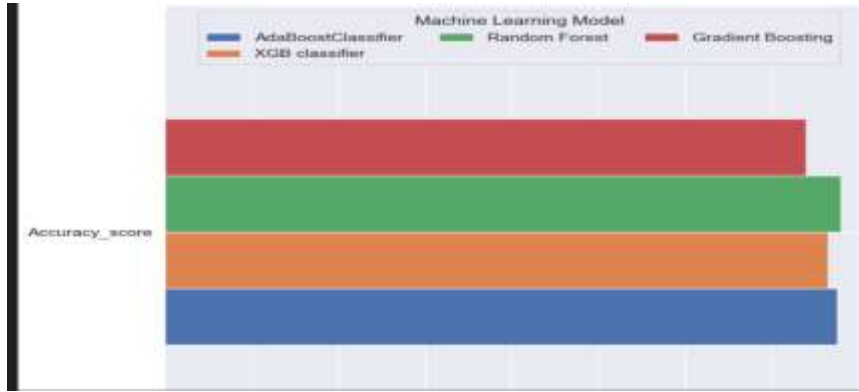
The Model Optimization and Tuning Phase involves refining neural network models for peak performance. It includes optimized model code, fine-tuning hyperparameters, comparing performance metrics, and justifying the final model selection for enhanced predictive accuracy and efficiency.

Hyperparameter Tuning Documentation (8 Marks):

Model	Tuned Hyperparameters
Gradient Boosting Classifier	<p>#importing the library for grid search <code>from sklearn.model_selection import GridSearchCV</code></p> <p>The 'lr_param_grid' specifies different values for regularization strength (C), solvers (solver), and penalty types (penalty). GridSearchCV (lr_cv) is employed with 5-fold cross-validation (cv=5), evaluating model performance based on accuracy (scoring="accuracy"). The process uses all available CPU cores (n_jobs=-1) for parallel processing and provides verbose output (verbose=True) to track progress.</p> <pre> from sklearn.model_selection import GridSearchCV from sklearn.ensemble import GradientBoostingClassifier rf = GradientBoostingClassifier() param_grid = {'n_estimators': [100, 200, 300], 'criterion': ['entropy', 'squared_error'], 'max_depth': [10, 20, 30], 'max_features': ['auto', 'sqrt']} grid_search = GridSearchCV(rf, param_grid, cv=5, n_jobs=-1, verbose=0) grid_search.fit(X_train, y_train) </pre> <p>✓ 33s</p> <p>Fitting 5 folds for each of 36 candidates, totalling 180 fits</p> 
Random Forest	<p>The parameter grid (rfc_param_grid) for hyperparameter tuning. It specifies different values for the number of trees (n_estimators), splitting criterion (criterion), maximum depth of trees (max_depth), and maximum number of features considered for splitting (max_features). GridSearchCV (rfc_cv) is employed with 3-fold cross-validation (cv=3), evaluating model performance based on accuracy (scoring="accuracy").</p> <pre> from sklearn.model_selection import GridSearchCV from sklearn.ensemble import RandomForestClassifier rf = RandomForestClassifier() param_grid = {'n_estimators': [100, 200, 300], 'criterion': ['entropy', 'gini'], 'max_depth': [10, 20, 30], 'max_features': ['auto', 'sqrt']} grid_search = GridSearchCV(rf, param_grid, cv=3, n_jobs=-1, verbose=0) grid_search.fit(X_train, y_train) </pre> <p>✓ 44s</p> <p>Fitting 3 folds for each of 36 candidates, totalling 108 fits</p> 

<p>XGBoost Classifier</p>	<p>The (params) define a grid for hyperparameter tuning of the XGBoost Classifier (XGBClassifier), including min_child_weight, gamma, colsample_bytree, and max_depth. The XGBClassifier is configured with a learning rate of 0.5, 100 estimators, using a binary logistic regression objective, and utilizing 3 threads for processing. GridSearchCV (xg_cv) is used with 5-fold cross-validation (cv=5), refitting the best model (refit=True), evaluating based on accuracy (scoring="accuracy")</p> <pre data-bbox="438 514 1214 745"> from sklearn.model_selection import GridSearchCV from xgboost import XGBClassifier rf = RandomForestClassifier() param_grid = {'n_estimators': [100, 200, 500], 'criterion': ['entropy', 'gini'], 'max_depth': [10, 20, 30], 'max_features': ['auto', 'sqrt']} grid_search = GridSearchCV(rf, param_grid, cv=5, n_jobs=-1, verbose=0) grid_search.fit(X_train, y_train) # fit fitting 5 folds for each of 16 candidates, totalling 80 fits = GridSearchCV (0.0) = best_estimator_: XGBClassifier = XGBClassifier </pre>
---------------------------	---

Final Model Selection Justification (2 Marks):

Final Model	Reasoning
Random Forest	<p>Random Forest model is chosen for its robustness in handling complex datasets and its ability to mitigate overfitting while providing high predictive accuracy.</p>  <p>Above all the models Random Forest model have the highest accuracy among all the models.</p>