**Binary and Multiclass Classification:**

**A Comparative Study on Breast Cancer Diagnosis and Data Science Job Experience Prediction**

Junyu Li and Lanjun Wang

CSC272: Machine Learning

May 14, 2025

**Abstract** - This project explores the application of supervised machine learning to two tasks: breast cancer diagnosis (binary classification) and job experience prediction (multiclass classification). We evaluated multiple models, including logistic regression, decision trees, random forests, and XGBoost, using cross-validation and hyperparameter tuning. For the binary task, recall and ROC AUC were prioritized to reduce false negatives. For the multiclass task, macro F1 and accuracy were used to address class imbalance. Results highlight the importance of aligning model complexity with data characteristics and show that simpler models often outperform more complex ones in real-world scenarios.

**Introduction**

Supervised machine learning plays a crucial role in addressing classification problems across domains, from healthcare to human resources. This project explores two distinct tasks using supervised learning: diagnosing malignant tumors from medical data (binary classification) and predicting experience levels in data science roles based on job-related attributes (multiclass classification). Each task presents unique challenges—such as class imbalance, overlapping features, and heterogeneous input types—that require thoughtful model selection and metric design. Our objective is to compare a variety of algorithms under a consistent experimental framework, assess their performance using task-appropriate metrics, and draw insights into how different modeling approaches respond to the nature of each dataset.

**Datasets**

*Dataset 1: Breast Cancer Wisconsin Dataset (Binary Classification)*

The binary classification task uses the Breast Cancer Wisconsin (Diagnostic) dataset, which contains 455 samples and 32 numerical features derived from digitized images of fine needle aspirate (FNA) tests of breast masses. The target variable is diagnosis, encoded as 0 (benign) and 1 (malignant). The dataset has no missing values and exhibits a moderate class imbalance, with benign samples slightly more common than malignant.

To ensure reliable evaluation, the dataset was randomly split into training (80%) and testing (20%) subsets while preserving class ratios (stratified split). Feature scaling using standardization was applied to the training set to account for variation in measurement scales and improve model convergence.

Exploratory data analysis revealed that while some features (e.g., mean radius, mean texture) help differentiate between classes, there is substantial overlap in feature distributions. Additionally, the correlation heatmap showed that some features are highly collinear, while others are weakly related or redundant. These challenges make classification more difficult, especially for linear models.

*Dataset 2 – Data Science Job Salaries Dataset (Multiclass Classification)*

The multiclass classification task uses the Data Science Job Salaries dataset, which contains 606 records and 12 features. The goal is to predict the experience\_level of a data science professional, categorized into four classes: Entry-level (EN), Mid-level (MI), Senior-level (SE), and Executive (EX).

The dataset includes a mix of feature types: 5 categorical, 1 ordinal, 5 numerical, and 1 binary. Before modeling, categorical features were one-hot encoded and ordinal features were label-encoded. The dataset was split intro training (79%) and testing (21%) sets, and standard scaling was applied to the numerical features. No missing values were present.

A key challenge with this dataset is the weak correlation between most features and the target variable. Additionally, the class distribution is imbalanced, with senior-level positions compressing nearly half of the data and executive-level positions making up less than 5%. Some categorical features, such as job\_title, introduce semantic ambiguity. As different titles may reflect similar roles. These factors complicate model training and reduce predictive performance, particularly for underrepresented classes.

**Methods**

***Binary Classification***

We selected four supervised learning algorithms to model the binary classification task:

* Logistic Regression
* Support Vector Machine
* Random Forest Classifier
* XGBoost Classifier

These models were chosen to balance interpretability, robustness to noise, and performance on structured tabular data. Logistic regression serves as baseline linear model, while SVM is well-suited for high-dimensional features spaces. Random Forest and XGBoost, both ensemble methods, are expected to handle feature redundancy and class overlap more effectively.

Each model was integreated intro a pipeline that included StandardScaler for feature normalization. Hyperparameter tuning was performed using GridSearchCV with 3-fold cross-validation. For example, logistic regress was tuned on the regularization strength C and penalty type (L1, L2), while XGboost was tuned on parameters such max\_depth, learning\_rate, and regularization terms (reg\_alpha, reg\_lambda).

To evaluate performance, we prioritized recall and ROC AUC due to the medical context, where false negatives (missing malignant cases) are particularly costly. Recall ensures we capture the majority of positive cases, while ROC AUC provides a threshold-independent view of classification performance.

***Multiclass Classification***

For the multiclass classification task, the following four models were used:

* Decision Tree Classifier
* K-Nearest Neighbors
* Random Forest Classifier
* XGBoost Classifier

These models were selected to explore how distance-based, tree-based, and ensemble methods perform on a feature-diverse and moderately imbalanced dataset. Decision Trees and Random Forests can handle categorical data and non-linear interactions, while KNN captures local patterns. XGBoost was included for its strong performance on structured classification tasks and its built-in support for handling class imbalance.

Categorical variables were encoded using one-hot encoding, and numerical features were standardized using StandardScaler. Hyperparameter tuning was again conducted using GridSearchCV, focusing on model-specific parameters such as tree depth, number of neighbors, or learning rate.

Performance was evaluated using macro-averaged F1 score and accuracy. Macro F1 treats each class equally, making it suitable for imbalanced classification, while accuracy provides an overall performance reference. Together, these metrics offer a balanced view of model effectiveness across all experience levels.

**Hyperparameter Tuning**

*Binary Classification*

***Logistic Regression***

The Logistic Regression model was tuned across a range of regularization strengths (C ∈ {0.01, 0.1, 1, 10}) and penalty types (l1, l2). The best configuration was C = 1 with L2 regularization, achieving a cross-validated recall of 0.9588. As shown in the tuning plot (Figure 1), L2 regularization consistently outperformed L1, especially at moderate C values. Performance A graph of a graph with lines and numbers

AI-generated content may be incorrect.decreased at extreme values, likely due to under- or over-regularization. The tuning process was efficient, completing in 0.12 seconds.

Figure 1 – Logistic Regression Tuning Curves for Different Penalties and C values

This indicates that a moderate amount of regularization yields optimal sensitivity for this dataset.

**A graph with a line

AI-generated content may be incorrect.*Support Vector Machine***

Figure 2 - SVM Recall across Different Values of C (Linear Kernal)

SVM was tuned over three hyperparameters: C ∈ {0.1, 1, 10}, kernel ∈ {'linear', 'rbf'}, and gamma ∈ {'scale', 'auto'}. The best-performing configuration used a linear kernel with C = 1 and gamma = scale, producing a cross-validated recall of 0.9529.

A graph of different colored squares

AI-generated content may be incorrect.The tuning curve (Figure 2) shows that linear kernels performed consistently well, while higher C values slightly decreased recall, suggesting mild overfitting. The RBF kernel did not offer improvements, indicating that the dataset is linearly separable. The tuning process completed in 0.34 seconds, highlighting SVM’s computational efficiency on moderate-sized datasets.

***Random Forest***

Random Forest was tuned on the number of trees (n\_estimators ∈ {50, 100, 200}) and maximum tree depth (max\_depth ∈ {None, 5, 10}). The optimal setting was n\_estimators = 50 with max\_depth = None, which achieved a cross-validated recall of 0.9294.

The heatmap (Figure 3) shows that unrestricted depth yielded the highest recall across all values of n\_estimators. Shallower trees underperformed, suggesting that complex feature interactions are beneficial for this dataset. However, deeper trees may risk overfitting if not managed carefully. Tuning completed in 3.23 seconds.

Figure 3 - Random Forest Heatmap Showing Recall by Depth and Number of Trees

***XGBoost***

A graph of a number of blue squares

AI-generated content may be incorrect.XGBoost was tuned over a wide range of hyperparameters, including learning rate ({0.01, 0.1, 0.2}), tree depth ({3, 5, 7}), and regularization parameters (reg\_alpha ∈ {0, 0.1}, reg\_lambda ∈ {1, 10}). The best configuration was found to be learning\_rate = 0.1, max\_depth = 3, reg\_alpha = 0, and reg\_lambda = 1, yielding a cross-validated recall of 0.9471. The tuning process required 10.77 seconds, the longest among the four models.

As shown in Figure 4, recall significantly improved as the learning rate increased from 0.01 to 0.1, with diminishing returns beyond that. Shallower and moderately deep trees (depth = 3 to 5) performed best, with a maximum recall of 0.941 achieved at learning\_rate = 0.2 and max\_depth = 5. In contrast, very shallow trees with low learning rates consistently underperformed. These results indicate that XGBoost benefits from moderate learning and structural complexity while avoiding overfitting through its regularization capabilities.

Figure 4 - XGBoost Tuning Heatmap Showing Recall by Learning Rate and Tree Depth

A screenshot of a graph

AI-generated content may be incorrect.*Multiclass Classification*

***Decision Tree***

The Decision Tree model was tuned over two hyperparameters: max\_depth ∈ {3, 5, 10} and min\_samples\_split ∈ {2, 5, 10}. The best configuration was max\_depth=10 and min\_samples\_split=2, which achieved the highest cross-validated F1 macro score of 0.412.

As shown in Figure 5, performance generally improved with deeper trees, especially when max\_depth=10. Models with smaller min\_samples\_split values performed slightly better, likely because they allowed deeper, more granular splits. Very shallow trees (e.g., max\_depth=3) underperformed, plateauing around an F1 of 0.377. Tuning completed in 1.81 seconds, indicating low computational cost for tree-based models.

Figure 5 - Decision Tree F1 Marco Heatmap

**A blue squares with numbers

AI-generated content may be incorrect.*KNN***

Figure 6 - KNN F1 Marco Heatmap

KNN was tuned across n\_neighbors ∈ {3, 5, 7} and weights ∈ {'uniform', 'distance'}. The best configuration was n\_neighbors=5 with weights='distance', yielding a cross-validated F1 macro score of 0.473, the highest among all models.

As shown in Figure 6, using distance-based weighting consistently outperformed uniform weights. The F1 score peaked at 0.473 for n\_neighbors=5, with performance declining slightly for both smaller (k=3) and larger (k=7) neighbor counts. KNN tuning was extremely efficient, finishing in 0.14 seconds.

**Random Forest**

A graph of different colors

AI-generated content may be incorrect.Random Forest was tuned over n\_estimators ∈ {50, 100} and max\_depth ∈ {10, 20, None}. The best performance (F1 macro = 0.433) was achieved with n\_estimators=100 and max\_depth=None, as shown in Figure 7.

Figure 7 - Random Forest F1 Macro Heatmap

The heatmap reveals that unrestricted depth consistently outperformed fixed depths. Increasing the number of trees from 50 to 100 also improved F1 across all depths. Shallow trees (e.g., max\_depth=10) led to degraded performance (F1 as low as 0.370), highlighting the value of deep, complex ensembles in this task. The tuning process required 0.42 seconds.

**A chart of a heatmap

AI-generated content may be incorrect.XGBoost**

Figure 8 - XGBoost F1 Macro Heatmap

XGBoost was tuned over learning\_rate ∈ {0.05, 0.1} and max\_depth ∈ {3, 6}. The optimal configuration was learning\_rate=0.05, max\_depth=6, which achieved the highest F1 macro score of 0.504—the best overall.

Figure 8 illustrates that deeper tree (depth=6) and smaller learning rates consistently outperformed other combinations. A learning rate of 0.05 paired with higher depth provided a strong balance between learning capacity and generalization. XGBoost tuning was completed in 0.86 seconds. However, due to duplicate parameter entries in the grid, the plotting initially failed until the parameter grid was cleaned.

**Result**

*Binary Classification*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Model | Hyperparameters | Tuning Time (sec) | Test Recall (Class 1) | Test ROC AUC | Test Accuracy |
| Logistic Regression | C=1, penalty='l2' | 0.12 | 0.98 | 0.9997 | 0.99 |
| SVM | C=1, kernel='linear', gamma='scale' | 0.34 | 0.98 | 0.9997 | 0.98 |
| Random Forest | n\_estimators=50, max\_depth=None | 3.23 | 0.93 | 0.9950 | 0.96 |
| XGBoost | learning\_rate=0.1, max\_depth=3, reg\_alpha=0, reg\_lambda=1 | 10.77 | 0.90 | 0.9970 | 0.96 |

*Multiclass Classification*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Model | Hyperparameters | Tuning Time (sec) | Test Macro F1 | Test Accuracy |
| Decision Tree | max\_depth=10, min\_samples\_split=2 | 1.81 | 0.4835 | 0.5859 |
| KNN | n\_neighbors=5, weights='distance' | 0.14 | 0.5685 | 0.6172 |
| Random Forest | n\_estimators=100, max\_depth=None | 0.42 | 0.5465 | 0.5938 |
| XGBoost | learning\_rate=0.05, max\_depth=6, reg\_alpha=0, reg\_lambda=1 | 0.86 | 0.3989 | 0.5938 |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | Class | | | | | | | |
|  | 0 (Support 19) | | 1 (Support 45) | | 2 (Support 59) | | 3 (Support 5) | |
| Model | Recall | Precision | Recall | Precision | Recall | Precision | Recall | Precision |
| Decision Tree | 0.16 | 0.50 | 0.42 | 0.61 | 0.86 | 0.58 | 0.40 | 0.67 |
| KNN | 0.37 | 0.47 | 0.62 | 0.58 | 0.71 | 0.67 | 0.40 | 1.00 |
| Random Forest | 0.37 | 0.44 | 0.49 | 0.54 | 0.76 | 0.65 | 0.40 | 1.00 |
| XGBoost | 0.26 | 0.50 | 0.53 | 0.60 | 0.80 | 0.60 | 0.00 | 0.00 |

**Discussion**

***Binary Classification***

The binary classification task focused on predicting whether a breast tumor is benign or malignant, using the Breast Cancer Wisconsin dataset. Given the medical context, recall for class 1 (malignant tumors) was prioritized to minimize the risk of false negatives, which are particularly harmful in diagnostic settings.

Across all four models, performance on the test set was strong, with Logistic Regression and SVM emerging as top performers. Both achieved a recall of 0.98, a ROC AUC of 0.9997, and overall accuracy above 0.98. These results indicate that the decision boundary separating benign and malignant cases is largely linear, which explains why these relatively simple models performed as well as — or better than — more complex ensemble methods. Additionally, their short training times and interpretability make them strong candidates for deployment in real-world screening tools.

Ensemble models like Random Forest and XGBoost also demonstrated high overall performance, but with slightly lower recall (0.93 and 0.90, respectively). Despite achieving high ROC AUCs (0.9950 and 0.9970), their tendency to misclassify a few malignant cases reflects either increased model variance or conservative thresholding. XGBoost, in particular, showed robust ranking ability but less aggressive positive classification, suggesting it was more cautious in labeling tumors as malignant. This behavior may be beneficial in some settings but less desirable when missing malignant cases is costly.

Hyperparameter tuning revealed that linear models required minimal configuration — a moderate regularization (C=1) for Logistic Regression and linear kernel SVMs were sufficient. Ensemble methods benefited from deeper exploration: XGBoost performed best with shallow trees (max\_depth=3) and a learning rate of 0.1, while Random Forest favored more trees with unrestricted depth. Tuning times reflected this complexity, ranging from under 1 second for linear models to over 10 seconds for XGBoost.

In summary, simple linear classifiers not only generalized better but also aligned more directly with the primary goal of capturing all malignant cases. Ensemble models, while still strong, did not offer additional benefit in this context and were more computationally demanding. These findings reinforce the value of model simplicity and sensitivity-first evaluation when designing machine learning solutions for high-risk domains like healthcare.

***Multiclass Classification***

In the multiclass classification task, we aimed to predict data science job experience levels using models trained on a heterogeneous feature set. The primary evaluation metrics were macro-averaged F1 score, which ensures equal importance across all classes, and accuracy, which provides a general view of correctness. These metrics were chosen in Phase 2 to account for class imbalance and diverse performance expectations across experience levels.

Among the four models evaluated, K-Nearest Neighbors (KNN) emerged as the most balanced performer. With a test macro F1 of 0.5685 and accuracy of 61.72%, KNN successfully captured patterns across all classes, including the underrepresented Executive level (class 3). Its distance-weighted approach helped reduce bias toward majority classes and maintained flexibility without overfitting. KNN also had the shortest tuning time (0.14s), making it both accurate and efficient.

Random Forest achieved slightly lower scores than KNN, with a macro F1 of 0.5465 and accuracy of 59.38%. It demonstrated strong generalization, particularly for class 2 (Senior), but its performance declined on minority classes. The model benefitted from ensemble diversity and deep trees, but was less responsive to minority class variation, possibly due to the dataset’s limited size.

The Decision Tree classifier, though simple and interpretable, yielded weaker results (macro F1 = 0.4835, accuracy = 58.59%). It showed sensitivity to the dominant classes but lacked the depth and generalization capacity of ensemble or distance-based methods. Its relatively high tuning time (1.81s) further highlighted the tradeoff between model flexibility and performance in low-data settings.

Surprisingly, XGBoost, which had the highest training F1 and most advanced hyperparameter tuning, delivered the lowest test macro F1 (0.3989). While its accuracy (59.38%) remained comparable to other models, XGBoost completely failed to identify class 3 (Executive), yielding zero recall and F1 for that class. This suggests overfitting to frequent patterns in the training data and an inability to generalize to rare classes. Despite moderate tuning time (0.86s), its poor minority class performance limits its utility in this context.

Overall, while all models performed comparably in terms of accuracy (~58–62%), macro F1 scores revealed more nuance in class-wise generalization. KNN offered the best tradeoff between accuracy, fairness across classes, and computational efficiency. Random Forest was also strong but less responsive to minority classes. XGBoost, though often high-performing in other contexts, was least effective here due to overfitting and class imbalance sensitivity.

**Conclusion**

This project demonstrated that model effectiveness depends heavily on task type and dataset characteristics. For binary classification, simple linear models like logistic regression and SVM delivered outstanding results with high recall and minimal complexity, making them suitable for sensitive applications like medical diagnosis. In contrast, the multiclass classification task was more challenging due to class imbalance and weak feature correlations. KNN emerged as the most balanced performer, offering strong generalization across all classes, while XGBoost, despite its training strength, failed to predict minority classes effectively.

With more time and resources, we would have explored advanced techniques for class imbalance (e.g., SMOTE or class weighting), deeper feature engineering, and larger datasets to improve generalization. Future directions include experimenting with ensemble stacking, automated hyperparameter tuning, and applying model interpretability tools like SHAP to better understand and trust model decisions.

**Reference**

1. OpenAI. (2024). ChatGPT (GPT-4 model). https://chat.openai.com
2. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., ... & Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. Journal of Machine Learning Research, 12, 2825–2830. https://scikit-learn.org
3. Chen, T., & Guestrin, C. (2016). XGBoost: A scalable tree boosting system. Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 785–794. https://xgboost.readthedocs.io
4. Joblib Developers. (2024). Joblib: Running Python functions as pipeline jobs. https://joblib.readthedocs.io
5. Hunter, J. D. (2007). Matplotlib: A 2D graphics environment. Computing in Science & Engineering, 9(3), 90–95. https://matplotlib.org