Diabetes Prediction Using the Pima Indians Diabetes Database

About The Project

In this project, I've used the Pima Indians Diabetes Database from Kaggle to predict diabetes onset and identify the most effective classification algorithm for this dataset. The dataset includes 768 records with 8 features (Pregnancies, Glucose, etc.) and a binary "Outcome" (~35% positive cases). Several challenges were identified:

1. **Missing Values**: Addressed with KNN imputation.
2. **Class Imbalance:** Applied SMOTE to balance the training set.
3. **Feature Scaling:** Standardized features for consistency.
4. **Overfitting/Underfitting:** Logistic Regression and Random Forest overfit (mitigated with regularization); XGBoost underfit (improved by tuning).
5. **Performance Limitations:** The dataset's size and class imbalance required careful optimization to ensure efficient training.

Models (Logistic Regression, Decision Tree, XGBoost, Random Forest) were trained on a 70% train, 15% CV, 15% test split. To tackle the challenges:

1. **Challenges 1 & 2 (Missing Values & Scaling):** Used KNN imputation and standardization.

2. **Challenge 3 (Class Imbalance)**:** Applied SMOTE to oversample the minority class, avoiding loss of predictive information.

3. **Challenge 4 (Overfitting/Underfitting):** Regularization and hyperparameter tuning were used to balance model performance.

Next Steps

The models have room for improvement:

1. Explore cost-sensitive learning or ADASYN as alternatives to SMOTE.
2. Optimize learning rate schedules to improve training stability.
3. Tune thresholds to maximize F1-scores.
4. Add interaction terms (e.g., BMI * Age) for better feature representation.
5. Explore stacking for improved accuracy.
6. Test on other datasets for generalizability.

Model Performance Comparison

Model	Split	Accuracy	Precision	Recall	F1-score	AUC
Logistic Regression	Train	0.78	0.74	0.60	0.66	0.85
	CV	0.74	0.59	0.46	0.52	0.82
	Test	0.74	0.73	0.50	0.59	0.85
Decision Tree	Train	0.80	0.79	0.59	0.67	0.87
	CV	0.76	0.62	0.51	0.56	0.79
	Test	0.75	0.80	0.45	0.58	0.82
XGBOOST	Train	0.75	0.60	0.83	0.70	0.85
	CV	0.69	0.49	0.77	0.60	0.80
	Test	0.71	0.58	0.82	0.68	0.81
Random Forest	Train	0.83	0.85	0.62	0.72	0.90
	CV	0.74	0.61	0.40	0.48	0.79
	Test	0.77	0.79	0.52	0.63	0.85

Comparison with Papers

- Hennebelle et al. (2023): On PIMA Indian, RF achieved 0.7827 accuracy (with feature selection); LR was ~0.7227 (RF outperformed by 6%). My RF (0.77) is slightly lower, but LR (0.74) outperforms their LR.
- Mousa et al. (2023): On PIMA Indian, LSTM got 0.85 accuracy (AUC=0.89), RF 0.78 (AUC=0.81), CNN 0.82 (AUC=0.86). My RF (0.77, AUC=0.85) is competitive with their RF but below LSTM and CNN. Deep learning models excel, but my models are simpler and interpretable.

Conclusion

My models achieved: LR (Recall=0.50, F1-Score=0.59), Decision Tree (Recall=0.45, F1-Score=0.58), XGBoost (Recall=0.82, F1-Score=0.68), and RF (Recall=0.52, F1-Score=0.63). Glucose was the top predictor (SHAP). XGBoost performed the best with the highest F1-Score (0.68) and Recall (0.82), making it particularly effective for imbalanced datasets by minimizing missed diabetes cases, which is critical in medical applications. Compared to literature, deep learning (e.g., LSTM: 0.85 accuracy) outperforms my models, but my RF and LR are competitive with simpler methods. SMOTE and tuning improved performance. Future work includes advanced ensembles and cost-sensitive learning.