Analysis of the 2010 American Community Survey Dataset:

Hiwa Tase

Machine Learning Course

1. Description of the Dataset

The dataset used in this analysis comes from the 2010 American Community Survey (ACS). It contains detailed information about 69,861 people, covering 16 different variables. Some of the key variables include health insurance status (nohealthins), marital status (marst), race (race2), total income (inctot), usual hours worked (uhrswork), and age. There are also additional columns that provide information about employment status, type of worker, education level, gender, and regional information. However, some variables, like WKSWORK2 and classwkr, have missing data.

This dataset was chosen because it provides a wide range of information that can help us understand many factors that influence whether someone has health insurance or not. Its broad scope makes it perfect for a detailed analysis of the social and economic factors that affect health insurance coverage. This comprehensive data allows us to explore how different aspects of people's lives impact their access to health insurance.

Prediction Goal

The primary objective of this analysis is to develop a predictive model that determines whether individuals have health insurance or not. This is represented by the binary variable "nohealthins," which indicates if a person does not have health insurance. The model will use various pieces of information about each person, including their age, marital status, race, total income, usual hours worked, employment status, type of job, education level, gender, and regional data. By analyzing these factors, we aim to identify the key determinants of health insurance status and accurately predict whether someone is likely to have health insurance. This model will provide valuable insights into the impact of different demographic and socioeconomic features on health insurance coverage.

2. Data Cleaning and Preprocessing

Initial Steps

The dataset was loaded and split into a training set (80%) and a testing set (20%) using stratified sampling to keep the health insurance status proportions consistent in both sets.

Cleaning the Data

Several steps were taken to clean and preprocess the data:

- **Handling Missing Values**: Columns with many missing values (WKSWORK2, classwkr, empstat, vetstat) were dropped.
- **Income Transformation**: Negative income values were removed, and incomes were capped at $500,000 to manage outliers.
- **Imputation**: Missing values in columns like inctot and age were filled in using the most common value.

Challenges

A major challenge was dealing with missing values in multiple columns, which required careful imputation to avoid bias. Additionally, transforming income data to handle outliers was crucial to ensure the models' accuracy and reliability.

3. Data Visualization Insights

Visualization was crucial for understanding the data distribution and identifying patterns.

Techniques Used

- **Histograms**: Used to display the distribution of key continuous variables such as inctot, uhrswork, and age.
- **Box Plots**: Provided insights into the distribution of these variables across the binary health insurance status.
- **Scatter Matrix**: Helped visualize correlations between multiple variables.

Findings

- **Income and Health Insurance**: Histograms and box plots revealed that higher income levels were generally associated with a higher likelihood of having health insurance.
- **Age Distribution**: Visualizations showed that age had a varied impact on health insurance status, with certain age groups more likely to be uninsured.
- **Work Hours**: The number of usual hours worked per week also showed a distinct pattern, with certain working hours associated with a higher probability of being uninsured.

Transformations

Further transformations were applied to the data to enhance model performance. For instance, squaring and logarithmic transformations of income and age helped normalize the distributions, which is beneficial for some machine learning algorithms.

4. Experiments with Supervised Learning Algorithms

K-Nearest Neighbors (KNN)

Description: KNN is a simple, instance-based learning algorithm that classifies new cases based on a majority vote of the k-nearest neighbors. It calculates the distance between the new case and existing cases using a specified metric (e.g., Euclidean or Manhattan distance) and assigns the most common class among the nearest neighbors.

Parameters Adjusted:

- n_neighbors: The number of neighbors to consider (tested values: 3, 5, 7, 9).
- weights: The weight function used in prediction (uniform or distance).
- metric: The distance metric to use (Euclidean or Manhattan).

Performance:

- **Default Parameters**: Precision: 0.7432, Recall: 0.7812, F1 Score: 0.7617.
- **Cross-Validation**: Precision: 0.7390, Recall: 0.7791, F1 Score: 0.7585.
- **Best Parameters (Grid Search)**: {'metric': 'manhattan', 'n_neighbors': 9, 'weights': 'uniform'}.
    - **Performance with Best Parameters**: Precision: 0.7516, Recall: 0.7902, F1 Score: 0.7704.

The KNN model's performance improved with the optimal parameters found through grid search, showing higher precision, recall, and F1 score. The choice of distance metric and number of neighbors significantly impacted the model's performance.

Random Forest Classifier

Description: Random Forest is an ensemble learning method that constructs multiple decision trees during training and outputs the mode of the classes for classification. It reduces overfitting by averaging multiple decision trees, which are trained on different parts of the data and different subsets of features.

Parameters Adjusted:

- n_estimators: The number of trees in the forest (tested values: 50, 100, 200).
- max_features: The number of features to consider when looking for the best split (auto, sqrt, log2).
- max_depth: The maximum depth of the tree (10, 20, 30, None).

Performance:

- **Default Parameters**: Precision: 0.7449, Recall: 0.7828, F1 Score: 0.7634.

- **Cross-Validation**: Precision: 0.7424, Recall: 0.7799, F1 Score: 0.7607.
- **Best Parameters (Grid Search)**: {'max_depth': 10, 'max_features': 'auto', 'n_estimators': 50}.
    - **Performance with Best Parameters**: Precision: 0.7396, Recall: 0.8573, F1 Score: 0.7941.

The Random Forest model showed a significant improvement in recall and F1 score with the optimal parameters. The ability to control overfitting through parameters like max_depth and the number of features considered at each split played a crucial role in enhancing the model's performance.

Comparison and Reflection

Both models showed stable performance with default and cross-validated parameters. However, the Random Forest model, with its ensemble approach, was more effective in capturing the underlying patterns in the data, as indicated by the higher recall and F1 score with optimized parameters. The KNN model's sensitivity to the choice of neighbors and distance metric highlights the importance of parameter tuning for achieving optimal performance.

5. Experiments with PCA for Feature Selection

Description: Principal Component Analysis (PCA) is a dimensionality reduction technique that transforms the data into a new coordinate system, where the greatest variances by any projection of the data lie on the first coordinates (principal components). By selecting a subset of these components, PCA can reduce the number of features while retaining the most important information.

Experiment and Results

PCA was applied to the training data to reduce its dimensionality before using it with the best-performing Random Forest model.

Steps:

1. Applied PCA to the training data and retained the principal components that explained 95% of the variance.
2. Trained the Random Forest model with the reduced dataset.
3. Evaluated the model's performance.

Performance:

- **Without PCA**: Precision: 0.7396, Recall: 0.8573, F1 Score: 0.7941.

- **With PCA**: Precision: 0.7321, Recall: 0.8450, F1 Score: 0.7854.

Using PCA for feature selection slightly reduced the model's performance in terms of precision, recall, and F1 score. While PCA effectively reduced the feature space, it also led to the loss of some information that was crucial for the Random Forest model to make accurate predictions.

Reflection

The application of PCA provided valuable insights into the trade-offs between dimensionality reduction and model performance. While PCA helps in reducing the computational complexity and mitigating the risk of overfitting, it is essential to balance this with the need to retain critical information for model accuracy. In this case, the slight drop in performance suggests that the original feature set contained important information that was not fully captured by the principal components.

Discussion on PCA as a Pre-processing Step for Clustering

Overview of PCA and Clustering Algorithms

Principal Component Analysis (PCA): PCA is a dimensionality reduction technique that transforms a dataset with many variables into a smaller set of uncorrelated variables, called principal components, while retaining most of the original variability in the data. The main goal of PCA is to simplify the dataset by reducing its dimensions, making it easier to visualize and analyze, and often speeding up subsequent machine learning tasks.

Clustering Algorithms:

1. **k-Means Clustering:** k-Means is an iterative algorithm that partitions the dataset into k clusters. Each cluster is represented by its centroid, which is the mean of the points within the cluster. The algorithm assigns each point to the nearest centroid, updates the centroids based on the assigned points, and repeats the process until convergence.
2. **Agglomerative Clustering:** This is a hierarchical clustering method that builds nested clusters by successively merging or splitting them. Agglomerative clustering starts with each point as a single cluster and merges the closest pairs of clusters until only the desired number of clusters remains.
3. **DBSCAN (Density-Based Spatial Clustering of Applications with Noise):** DBSCAN is a density-based clustering algorithm that groups together points that are closely packed together while marking points that lie alone in low-density regions as outliers. It relies on two parameters: epsilon (eps) for the neighborhood radius and the minimum number of points required to form a dense region.

Results of Using PCA for Clustering

Without PCA:

- **k-Means Clustering:** The algorithm identified clusters but with relatively low Adjusted Rand Index (ARI) and silhouette scores, indicating that the clusters did not align well with the true labels and were not very well-defined.
- **Agglomerative Clustering:** Similar to k-Means, the results showed low ARI and silhouette scores, reflecting poor cluster definition and alignment with true labels.
- **DBSCAN:** DBSCAN performed the best among the three methods without PCA, with higher silhouette scores suggesting well-defined clusters. However, the ARI was still low, indicating poor alignment with true labels.

With PCA:

- **k-Means Clustering:** The application of PCA slightly improved the performance of k-Means clustering. The ARI and silhouette scores were slightly higher compared to the results without PCA, indicating better-defined clusters and a closer alignment with true labels.
- **Agglomerative Clustering:** Similar to k-Means, PCA improved the performance of Agglomerative Clustering, with modest increases in ARI and silhouette scores.
- **DBSCAN:** The performance of DBSCAN was not significantly affected by PCA. The silhouette scores remained high, suggesting well-defined clusters, but the ARI stayed relatively low.

Insights and Conclusions

1. **Effectiveness of PCA:** PCA was effective in improving the performance of k-Means and Agglomerative Clustering by reducing the dimensionality of the data and eliminating noise. This resulted in better-defined clusters and slightly better alignment with the true labels.
2. **DBSCAN Robustness:** DBSCAN's performance remained robust even without PCA, indicating its strength in handling high-dimensional data and identifying well-defined clusters. This might be due to its ability to find arbitrarily shaped clusters and handle outliers effectively.

Summary:

Analyzing the 2010 American Community Survey dataset provided several key insights from data cleaning, preprocessing, model building, and feature selection. The projects showed the importance of thorough data cleaning, such as handling missing values and adjusting income data to manage outliers, which kept the data accurate and improved model performance. Supervised

learning algorithms like Random Forest and KNN performed well, with Random Forest being particularly good in recall and F1 score. Tuning parameters through grid search greatly improved model accuracy, especially for KNN. Data visualization methods like histograms, box plots, and scatter matrices helped identify patterns and relationships, showing that higher income levels and certain age groups significantly impacted health insurance status.

However, there were areas for improvement. Future projects could use better methods to fill in missing data, like machine learning algorithms. While PCA (Principal Component Analysis) helped reduce the number of variables, it also slightly reduced performance, indicating the need for a more balanced approach that combines PCA with selecting important features. Using more robust cross-validation techniques and trying other supervised algorithms like Gradient Boosting or Support Vector Machines could improve performance and offer better comparisons.

In conclusion, the projects highlighted the importance of thorough data preparation and the careful use of both supervised and unsupervised learning methods. The comprehensive dataset allowed for a detailed analysis of the factors affecting health insurance status, and effective data cleaning and preprocessing were crucial in building accurate predictive models. Future work can build on these insights by using advanced methods to handle missing data, balanced approaches to reduce variables, and a wider range of machine learning algorithms to further improve predictive accuracy and reliability.