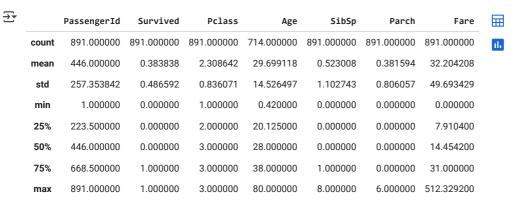
Machine Learning Lab

Bhuvnesh Verma

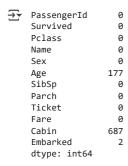
NAME :

data.describe()

```
# ROLL NO :
                 28
# SECTION :
                 Α
# LAB NO. :
                 03
                Data Preprocessing and Survival Prediction
Aim : To predict Titanic survivors using a Random Forest Classifier by preprocessing data,
engineering features, and applying PCA for dimensionality reduction.
# Load the Titanic dataset from a URL
url = "https://raw.githubusercontent.com/datasciencedojo/datasets/master/titanic.csv"
data = pd.read_csv(url)
# Display the first few rows of the dataset to understand its structure
print(data.head())
       PassengerId Survived Pclass \
                1
                           0
                 2
                 3
     3
                 4
                           1
                                   1
     4
                           0
                                   3
                                                            Sex
                                                                  Age SibSp \
                                                    Name
     0
                                Braund, Mr. Owen Harris
                                                           male 22.0
                                                                           1
       Cumings, Mrs. John Bradley (Florence Briggs Th...
     1
                                                          female
                                                                  38.0
                                                                           1
     2
                                  Heikkinen, Miss. Laina female
                                                                  26.0
                                                                           0
     3
            Futrelle, Mrs. Jacques Heath (Lily May Peel) female
                                                                  35.0
                                                                           1
     4
                                Allen, Mr. William Henry
       Parch
                                   Fare Cabin Embarked
                        Ticket
     0
                     A/5 21171
                                7.2500
                                          NaN
                                                     S
     1
                     PC 17599 71.2833
                                          C85
                                                     C
           0 STON/02. 3101282
                                 7.9250
                                          NaN
                        113803 53.1000 C123
     3
                                                     S
     4
                        373450
                                8.0500
                                          NaN
# Displays dataframe info and data types
data.info()
<class 'pandas.core.frame.DataFrame'>
     RangeIndex: 891 entries, 0 to 890
     Data columns (total 12 columns):
         Column
                      Non-Null Count Dtype
     ---
         PassengerId 891 non-null
                                      int64
     1
         Survived
                      891 non-null
                                      int64
                      891 non-null
                      891 non-null
         Name
                                      object
                      891 non-null
         Sex
                                      object
     5
         Age
                      714 non-null
                                      float64
         SibSp
                      891 non-null
                                      int64
                      891 non-null
                                      int64
         Parch
                      891 non-null
     8
         Ticket
                                      object
         Fare
                      891 non-null
                                      float64
     10 Cabin
                      204 non-null
                                      object
     11 Embarked
                      889 non-null
     dtypes: float64(2), int64(5), object(5)
     memory usage: 83.7+ KB
# Show summary statistics of numeric columns
```



Check for missing values in the dataset
print(data.isnull().sum())



Drop unnecessary columns that are not relevant for the analysis
data = data.drop(columns=["PassengerId", "Name", "Ticket", "Cabin", "Fare"])

Display the first few rows after dropping columns print(data.head())

∑ ₹		Survived	Pclass	Sex	Age	SibSp	Parch	Embarked
_	0	0	3	male	22.0	1	0	S
	1	1	1	female	38.0	1	0	C
	2	1	3	female	26.0	0	0	S
	3	1	1	female	35.0	1	0	S
	4	0	3	male	35.0	0	0	S

As there are missing value in Age cloumn
Fill missing values in the 'Age' column with the median value
data["Age"] = data["Age"].fillna(data["Age"].median())

As there are missing value in Embarked column. Use mode as we have only S & C # Fill missing values in the 'Embarked' column with the mode value data["Embarked"] = data["Embarked"].fillna(data["Embarked"].mode()[0])

Display the first few rows after filling missing values
print(data.head())

₹		Survived	Pclass	Sex	Age	SibSp	Parch	Embarked
	0	0	3	male	22.0	1	0	S
	1	1	1	female	38.0	1	0	C
	2	1	3	female	26.0	0	0	S
	3	1	1	female	35.0	1	0	S
	4	0	3	male	35.0	0	0	S

SibSp - the number of siblings and spouses a passenger had on board.

 $\mbox{\# Parch}$ - the number of parents and children a passenger had aboard the ship.

 $\mbox{\tt\#}$ Create a new feature 'family_size' by combining 'SibSp' and 'Parch' columns

data["family_size"] = data['SibSp'] + data['Parch'] + 1

Create a new feature 'Is_Alone' to indicate if a passenger is traveling alone
data["Is_Alone"] = (data["family_size"] == 1).astype(int)

Embarked - indicates the port of embarkation for each passenger. [C: Cherbourg , Q: Queenstown , S: Southampton]
Create a new feature 'Title' by combining 'Sex' and 'Embarked' columns
data["Title"] = data["Sex"] + "_" + data['Embarked']

Display the first few rows after creating new features print(data.head()) Sex Age SibSp Parch Embarked family_size \ _ Survived Pclass a 0 3 male 22.0 0 S 1 1 1 female 38.0 0 C 3 female 26.0 0 0 S 3 1 1 female 35.0 1 0 S 2 4 0 male 35.0 Title Is_Alone 0 male_S 0 0 female C 1 2 1 female_S 3 0 female_S 4 1 male_S # Convert categorical columns into binary columns using one-hot Convert to binary features called dummy variables 0: Observation was NOT that category 1: Observation was that category data = pd.get_dummies(data, columns=["Sex", "Embarked", "Title"], drop_first=True) # Drop 'SibSp' and 'Parch' columns as 'family_size' is a better representation data = data.drop(columns=["SibSp", "Parch"]) # Display the first few rows after removing SibSp & Parch print(data.head()) Survived Pclass Age family_size Is_Alone Sex_male Embarked_Q \ 0 0 3 22.0 2 0 True False 1 1 1 38.0 2 0 False False 2 1 3 26.0 1 False False 3 1 1 35.0 2 a False False 4 0 3 35.0 True False Embarked_S Title_female_Q Title_female_S Title_male_C Title_male_Q \ False 0 False False False True False False False False False 1 2 True False True False False True 3 True False False False 4 True False False False False Title_male_S 0 True 1 False False 3 False 4 True # Split the dataset into features (X) and target (y) X = data.drop(columns=["Survived"]) y = data["Survived"] # Standardize the features to have a mean of 0 and a standard deviation of 1 scaler = StandardScaler() X_scaled = scaler.fit_transform(X) # Split the dataset into training and testing sets (80% training, 20% testing) X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42) # Apply Principal Component Analysis (PCA) to reduce dimensionality pca = PCA(n_components=12) X_train_pca = pca.fit_transform(X_train) X_test_pca = pca.fit_transform(X_test) # Print the explained variance ratio of each principal component print("Explained variance ratio", pca.explained_variance_ratio_)

```
# Initialize a Random Forest Classifier with a fixed random state for reproducibility
rf = RandomForestClassifier(random_state=42)
```

1.83105681e-03 7.66358339e-04 4.80432941e-04 3.02982410e-04 2.36117755e-04 3.48823904e-19 2.54973412e-20 0.00000000e+00]

Explained variance ratio [9.79021339e-01 1.04195229e-02 4.17053134e-03 2.77165865e-03

```
# Train the Random Forest model on the training data
rf.fit(X_train, y_train)
            RandomForestClassifier
     RandomForestClassifier(random_state=42)
# Predict the target values for the test set
y_pred = rf.predict(X_test)
# Calculate the accuracy of the model
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy :", accuracy)
Accuracy: 0.8100558659217877
# Calculate the Mean Squared Error (MSE) and R<sup>2</sup> Score to evaluate the model
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)
print(f"Mean Squared Error : {mse}")
print(f"R2 Score : {r2}")
Mean Squared Error: 0.18994413407821228
     R<sup>2</sup> Score: 0.2167310167310167
```

Here is a concise explanation of PCA and Random Forest Classifier

1. Principal Component Analysis (PCA)

What is PCA?

- · A dimensionality reduction technique.
- Transforms original features into new uncorrelated features called **principal components**.

Why Use PCA?

- · Reduces dataset complexity.
- · Removes redundant or correlated features.
- Improves visualization of high-dimensional data.
- · Speeds up machine learning algorithms.

How PCA Works

- 1. **Standardize Data**: Scale features to have mean = 0 and standard deviation = 1.
- $2. \ \textbf{Compute Covariance Matrix}: \ \textbf{Measures how features vary together}.$
- Calculate Eigenvalues and Eigenvectors: Eigenvectors = directions (principal components), eigenvalues = variance along those
 directions.
- 4. Sort and Select Components: Keep top n components with the highest variance.
- 5. Transform Data: Project data onto the selected components.

Key Terms

- Principal Components: New uncorrelated features.
- Explained Variance Ratio: Proportion of variance captured by each component.
- Dimensionality Reduction: Reducing the number of features while retaining information.

Example Code

```
from sklearn.decomposition import PCA
pca = PCA(n_components=12)
X_train_pca = pca.fit_transform(X_train)
print("Explained variance ratio:", pca.explained_variance_ratio_)
```

2. Random Forest Classifier

What is Random Forest?

- An ensemble learning method for classification and regression.
- · Combines multiple decision trees to improve accuracy and reduce overfitting.

Why Use Random Forest?

- · High accuracy due to ensemble averaging.
- · Robust to overfitting.
- · Handles missing data and outliers.
- · Provides feature importance scores.

How Random Forest Works

- 1. Bootstrap Sampling: Create random subsets of the training data for each tree.
- 2. Feature Randomness: At each split, consider a random subset of features.
- 3. Build Decision Trees: Train multiple trees on different subsets.
- 4. Voting/Averaging: For classification, use majority voting; for regression, use averaging.

Key Terms

- Ensemble Learning: Combining multiple models for better performance.
- Decision Trees: Individual models that make predictions based on if-else conditions.
- Feature Importance: Scores indicating the contribution of each feature to predictions.

Example Code

```
from sklearn.ensemble import RandomForestClassifier
rf = RandomForestClassifier(random_state=42)
rf.fit(X_train, y_train)
y_pred = rf.predict(X_test)
print("Accuracy:", accuracy_score(y_test, y_pred))
```

Comparison: PCA vs Random Forest

Aspect	PCA	Random Forest
Purpose	Dimensionality reduction	Classification/Regression
Output	Transformed features (principal components)	Predictions (e.g., class labels or continuous values)
Key Idea	Reduce features while retaining variance	Combine multiple decision trees to improve accuracy
Use Case	Preprocessing step for high-dimensional data	Final model for making predictions
Interpretability	Principal components are not directly interpretable	Provides feature importance for interpretability
Handles Overfitting	Reduces overfitting by reducing dimensionality	Reduces overfitting through ensemble averaging

When to Use PCA and Random Forest Together?

- High-Dimensional Data: Use PCA to reduce features before training Random Forest.
- Improving Performance: PCA removes noise, which can improve Random Forest's accuracy.
- Visualization: PCA helps visualize data in 2D/3D, while Random Forest makes predictions.

Workflow of Practical

- 1. Load Data: Load dataset (e.g., Titanic dataset).
- 2. Preprocess Data: Handle missing values, encode categorical variables, and create new features.
- 3. Apply PCA: Reduce dimensionality using PCA.
- 4. Train Random Forest: Train the model on the reduced dataset.
- 5. Evaluate Model: Check accuracy, MSE, and R2 score.

```
Conclusion: The model achieved good accuracy, highlighting the importance of feature selection, data cleaning, and dimensionality reduction in improving prediction performance.
```