**Joining Datasets**

Set up 2 datasets designed to join together

A screenshot of a computer code

Description automatically generated 🡺 A screenshot of a phone

Description automatically generated

Process of joining datasets

1. Set index using ***.set\_index ( )***

A screenshot of a phone

Description automatically generated 🡺 by default, index is assumed for its natural index for each row.

* We can set the **IDs of each flower as index**
  + For flower name

A screenshot of a computer

Description automatically generated

* + For flower price

A screenshot of a computer

Description automatically generated

1. Join the dataset by set index using ***df.join ( )***

* **A.join(B) | A.join(B, how = ‘left’) 🡺 left join**

A screenshot of a computer

Description automatically generated 🡺 A diagram of a diagram

Description automatically generated

* + Left join is by default, meaning that we don’t need to add any other commands.

A screenshot of a computer

Description automatically generated

* + It will keep the rows from the left regardless of missing value on the right dataset.
* **A.join(B, how = ‘right’) 🡺 right join**

A screenshot of a computer

Description automatically generated 🡺 A diagram of a diagram

Description automatically generated with medium confidence

* + It will keep the rows from the right regardless of missing value on the left.
* **A.join(B, how = ‘inner’) 🡺 inner join**

A screenshot of a computer

Description automatically generated 🡺 A blue circle with black text

Description automatically generated

* + It will only show the info that 2 datasets both shares.
* **A.join(B, how = ‘outer’) 🡺 outer join**

A screenshot of a computer

Description automatically generated 🡺 A diagram of a diagram

Description automatically generated with medium confidence

* + It will include all of the information from both datasets into one.

1. For the dataset read csv file by Pandas, we can set the index using ***index\_col = ‘col’***

* Original dataset read by Pandas

A screenshot of a phone

Description automatically generated 🡺 natural index

* Convert the index as purchase\_id as **index\_col = ‘purchase\_id’**

A screenshot of a computer

Description automatically generated 🡺 assigned index

1. Join (***A.join(B)***) the dataset with its one-hot encoding by **pd.get\_dummies ( )**

* Specify a 2-D dataset with the column

A screenshot of a computer

Description automatically generated

* Convert this dataset with one-hot encoding

A screenshot of a computer

Description automatically generated 🡺 it will be a new dataset

* Use A.join(B) to join both

A screenshot of a computer

Description automatically generated

1. Join the columns using ***pd.merge (A, B)***

* This way we don’t need to set the new index by just assigning the common index in both datasets.
* We can join 2 datasets by assigning each column in the datasets

A screenshot of a computer code

Description automatically generated

* + It assigns joining task with left data and right data on the same column “flower\_id”.
  + Since both column indices are the same, it will operate as inner join.

A screenshot of a computer code

Description automatically generated

* We can join 2 datasets by assigning the joining method by **on = ‘index’, how = ‘command’**

A screenshot of a computer code

Description automatically generated

**Dimensionality Reduction**

Reducing dimension is to reduce the number of input features in a dataset while retaining as much important information as possible.

**Motivation of dimensionality reduction**

1. Having too many redundant features will result in model complexity with potential overfitting.
2. Too many features including irrelevant ones will decrease calculation speed.
3. Some unimportant features might be related to other features to make the model less interpretable.

**2 ways to reduce feature dimension**

* Feature selection: choose a subset of original features.
* Feature extraction: combine or transform the features to generate a new feature space.

**Dimension reduction in example**

1. Set the IVs (each column) and DV (target)

A close-up of a logo

Description automatically generated

1. Visualize the training X data

A screenshot of a computer

Description automatically generated

1. Implement train test split

* Turning multi-class classification into binary classification

A close up of a text

Description automatically generated

* Use stratified CV



1. Visualize statistics among training data

* We apply **.transpose()** to swap the rows and columns in the statistics dataset
* The mean and SD is highly skewed for normalization.

A screenshot of a computer

Description automatically generated

1. Standardize features by transposing mean and median among training data using ***.StandardScaler ( )***

* Standard Scaler will transform the training data and test data into z-score

A screenshot of a computer

Description automatically generated

* Prevent transformation data leakage
  + ***When transforming data, fit the training set, transform both train and test***

A diagram of a model

Description automatically generated

* + - By **fitting transformations** ***only on the training set*** and **carefully managing the train/test split**, you can reduce the risk of data leakage and ensure your model’s evaluation is reliable.

A screenshot of a computer code

Description automatically generated

* + ***When performing prediction, train on the training set, evaluate on the test set***

A diagram of a model

Description automatically generated

* + - Fit the model only with training data and predicting with test data

A screen shot of a computer code

Description automatically generated

**Feature Selection**

Based on how much each feature contribute to the prediction, we can select *the subset of entire feature* to reduce the dimensionality.

1. **Linear regression with Lasso Regulation**

* It can turn the unimportant features coefficient to exactly zero to shrink the dimensionality.

1. **Tree Based Model with Random Forest**

* It can visualize the feature importance of each feature and select the features that more relevant to the prediction.

1. **Univariate Test**

* A statistical method to assess the relationship between each feature and the target variable independently (one variable at a time).

1. **Recursive Feature Selection**

* An iterative method that selects features by recursively training a model and removing the least important features based on their importance scores.

**Lasso with *(C and L1 norm):***

* Lasso drives the coefficient of uninformative features to 0.

Original logistic regression without penalty

A screenshot of a computer code

Description automatically generated

* some features might not be relevant to the final predictions, so we might need to turn them into 0 so that we can reduce the model complexity.

A screenshot of a computer

Description automatically generated

* + applying C and penalty = ‘l1’ will shrink the irrelevant features into 0.

**Tree Based model with Random Forest using *model.feature\_importances\_***

1. Trees split the features based on removing impurity
2. **We can rank the feature based on how much impurity they remove**

* Features with more impurity reduction are considered as more important to prediction, and vice versa.

1. We can get each feature importance to the model prediction

A screenshot of a computer code

Description automatically generated

1. We can visualize the feature importance of each feature with corresponding values

A graph with blue and white bars

Description automatically generated

**Univariate Test using *SelectKBest ( f\_classif )***

* To evaluate each feature individually to determine its relevance to the target variable, without considering interactions between features.
* This is useful for identifying features with the strongest individual relationships to the target.

1. **SelectKBest ( )**

A screenshot of a computer code

Description automatically generated

* Conduct F-test for each training features against the target variable using f\_classif
* Keep the most important 3 features based on the performance of F-test

1. **F-test** for ***numerical features with categorical targets***

* The F-test evaluates whether a feature has a significant linear relationship with the target variable by compares the variance between groups (explained variance) to the variance within groups (unexplained variance).
* A higher F-value indicates a *stronger relationship between the feature and the target*.

**Recursive Feature elimination using *RFE***

* To iteratively removes the least important features, reducing the feature set step by step, and evaluates the model’s performance at each step.

Process of RFE

1. Train the Model:

* Train a model that assigns importance scores to features by coefficients in linear models or feature importance in tree-based models.

1. Rank Features:

* Rank features based on their importance by coefficients or feature importance score.

1. Eliminate Least Important Features:

* Iteratively remove the least important features, as determined by the model.

1. Repeat:

* Continue the process until the desired number of features is retained.

A screenshot of a computer

Description automatically generated

**Adjusted R-sq for standardized model performance comparison**

1. **R^2**: it represents the ***proportion of variance in the target variable explained by the features.***
   * Adding features to a model always increases R^2, even if the features are not relevant.
2. **Adjusted R^2**: adding penalty to the R-sq
   * Penalizes the R^2 value for adding irrelevant features.
   * Provides a better metric for comparing models with different numbers of features.
   * When the variables in the one dataset are the subset of the variables in another dataset, we need to compare both performances based on adjusted R-sq.

* Translate the formula to coding

A screenshot of a math problem

Description automatically generated

Comparison between model with 3 features and model with all the features

* **For R-sq**

A screenshot of a computer code

Description automatically generated

* **For adjusted R-sq by adj\_r2**

A close up of text

Description automatically generated

* By adding the penalty in R-sq, we can tell the adding more features will penalize the feature performance.
* Adding more features means adding some potential irrelevant noise, using adjusted R-sq will indicate if the added features is relevant to the model prediction.
  + Since the adjusted R-sq with all features is larger than only 3 features, we can conclude adding more feature to this model is adding more explanatory variables safely.

**Feature extraction**

Feature extraction involves transforming the original features into a new set of features (feature space) that captures the most relevant information from the data. This process can reduce redundancy, improve model performance, and help with visualization.

**Purpose**:

* **Visualization**: Reduce high-dimensional data into 2D or 3D for plotting and understanding.
* **Compression**: Reduce storage or computational needs while preserving critical information.
* **Dimensionality Reduction**: Eliminate irrelevant or redundant features to simplify the model.

**How It Works**:

* Create a new feature space by combining or transforming existing features.
* Maintain as much information or variance as possible (unsupervised methods) or maximize the separation between classes (supervised methods).

**PCA**

PCA is a dimensionality reduction technique that transforms the data into a new feature space

* It focuses on the directions (principal components) that capture the most variance.
* It is unsupervised, meaning it does not consider labels.

1. **Variance as Information:**

* PCA assumes that directions of high variance in the data are the most informative.
* Find Principal Components:
  + Identify directions (linear combinations of features) that explain the most variance.
    - **First Component**: The direction that explains the maximum variance in the data.
    - **Second Component**: A direction perpendicular to the first, capturing the next highest variance.
    - **Subsequent Components**: Each new component is orthogonal to all previous components and captures the remaining variance.

1. **Combining Collinear Features:**

* Features that are highly correlated (collinear) can be combined into a single principal component, reducing redundancy.

1. **Projection**:

* Data is projected onto a subspace with fewer dimensions while preserving as much variance as possible.

**Process**

1. **Center the Data**:

* Subtract the mean of each feature to ensure the data is centered around zero.
* This step is critical to align PCA with the directions of maximum variance.

1. **Find Principal Components**:

* Identify directions (linear combinations of features) that explain the most variance.

1. **Repeat Until Maximum Components**:

* The number of components cannot exceed the number of original features.

1. **Projection**:

* Transform the data by projecting it onto the principal components.

PCA coding template

A screen shot of a computer code

Description automatically generated

To the wine example

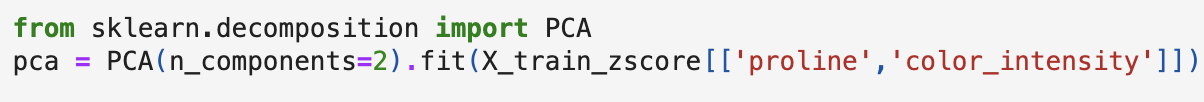
1. We can visualize how 2 features variance correlate each other without PCA

A screen shot of a graph

Description automatically generated

1. We can use PCA to remove the variance between 2 variables

* Perform PCA



* + The PCA object is created with n\_components=2, which means the data will be transformed into a space defined by the top two principal components.
  + fit calculates the principal components from the specified features proline and color\_intensity.
* Plot data with the highest PC



* + num\_components=1 suggests that the plot focuses on the first principal component (or components up to the specified number).

A graph with a line and a point

Description automatically generated with medium confidence

* plot data the first 2 highest PCs

A screenshot of a graph

Description automatically generated

**PCA in Explained Variance**

1. In the plot, we can tell the longer vector for each feature capture the higher variance of that feature.
2. In each repetition, each PCs will be shorter as each will explain less variance of its data.
3. We can visualize how each PC diminish each explained variance, indicating the each following PC will explain less than the previous ones.

A screen shot of a graph

Description automatically generated

* By capturing the Principal Components (PCs) from a subset of the original features, we can analyze how much variance each PC explains. This allows us to ensure that the selected subset of features retains enough information (e.g., 90% of the variance) compared to the original feature set of 1000 features.
  + From the plot, we can see the first 6 PCs have already capture 90% variance of the dataset where we can use the first 6 PCs for the later model performance.

**Purpose of PCA**

1. Instead of applying PCA to all 1000 features, you focus on a subset (e.g., proline and color\_intensity) for dimensionality reduction.
2. The goal is ***to verify whether the subset’s PCs can preserve the majority of the variance from the original feature set.***
3. Variance Explained by PCs:

* Each PC explains a portion of the total variance in the data.
* By summing the variance explained by the top PCs (e.g., the first few components), you can determine how much information is preserved.

1. Threshold for Information Preservation:

* Set a threshold (e.g., 90% variance) to decide if the subset is sufficient:
  + If the PCs from the subset explain ≥ 90% of the variance, the subset is likely adequate.
  + If not, additional features or components might be needed.