**Data Science & Machine Learning**

**Notebook**

**Professor: Andi Cupallari**

**Editor: Jackson Zhao**

**UNI: zz3119**

**Editing Time: 2024-2025 Fall Semester**

**Columbia University in the City of New York**

**Content**

**Machine Learning Model**

Supervised Learning Framework

Unsupervised Learning Framework

**Supervised Learning Domain**

Prediction and Interpretation

Supervised Learning Workflow

Data Splitting

Model Hyperparameter

Bias-Variance Tradeoff

Ways to prevent overfitting and underfitting

**Model Evaluation**

Metrics for Classification

Confusion matrix

Accuracy

Precision

Recall

Threshold for Precision and Recall

F-1 Score

ROC

R-sq

Adjusted R-sq

MSE

**Data Preprocessing: Imbalanced Dataset**

Stratified sampling

Random Undersampling with Majority Class

Random Oversampling with Minority Class

SMOTE

ADASYN

**Data Cleaning**

Duplicated Data

Missing Data

Feature Rescaling

Dealing with Skewness

Detecting and Removing Outliers

**Featuring Engineering**

Binning

One-hot Encoding

**Join the Data**

Method 1: df.join()

Method 2: pd.merge(A, B)

**Natural Language Processing**

Tokenization

Bag of Words (BoW)

n-Grams

Term Frequency

Document Frequency

Stopwords

TF-IDF

Topic Modeling

Latent Dirichlet Allocation (LDA)

**Supervised Machine Learning Model**

Regression Model

* Simple Linear Regression Model
* Ordinary Least Squares (OLS)
* Gradient Descent
* Multiple Linear Regression

Regularization

* Ridge Regression
* Lasso Regression
* Elastic-Net Regression

**Regression / Classification Model**

Logistic Regression

K-Nearest Neighbors Model

Decision Tree Model

Ensemble Method

* Bagging
* Random Forest Model
* Gradient Boosting Tree Model

Model Staking Technique

Perceptron and Multilayer Perceptron

Multiclass, Multilabel classification model

**Dimensionality Reduction**

Feature selection

* Linear Regression with Lasso Regularization
* Random Forest with Tree-based model.
* Univariate Test
* Recursive Feature Elimination

Feature Extraction

* PCA

**Unsupervised Learning Framework**

K-Means Clustering

Hierarchical Agglomerative Clustering

**Recommendation Engines**

Content-Based Filtering

Collaborative Filtering

Issues on Collaborative Filtering

**Model Pipeline**

Components of a pipeline

Column Transformer

**Machine Learning Model**

Machine learning model is used to perform several tasks based on the input features and output labels. These tasks include:

1. **Classification**: learn from existing columns to predict target class.
2. **Regression**: predict numerical values based on other columns.
3. **Interpretation**: explain relationship between features and prediction.
4. **Feature selection**: extract the important features from the whole columns.
5. **Clustering**: observation group feature space.

We use different machine learning models based on different nature of data and domain knowledge to interact between feature and target by 2 frameworks:

**Supervised Learning Framework**

* Supervised learning is a type of machine learning where model learns a function that *maps inputs based on sets of labeled training data*.
  + The training data contains input-output pairs where **input is the feature set** and **output the target variable**.
  + The goal is to make prediction based on training data and generalize well on unseen data to ensure accurate prediction and classification.
* Supervised learning often involves 2 main tasks
  + **Classification**: where model predicts a categorical label.
  + **Regression**: where model predicts continuous value.
* Common algorithms with supervised learning
  + Linear regression
  + Logistic regression
  + Support vector machine
  + Random Forest
  + Neural Network

**Unsupervised Learning Framework**

* It renders that model is *trained on unlabeled data*, meaning no target variables.
* The goal is to discover hidden patterns, relationships, or groupings within data without prior knowledge of outcomes.
* 2 main tasks for unsupervised learning
  + **Clustering**: the model aims to group data into clusters based on similarity
  + **Dimensionality reduction**: reduce # of irrelevant features while preserving maximum information.
* Common Algorithms
  + K-Means Clustering
  + Hierarchical Clustering
  + PCA

**Supervised Learning Domain**

**Prediction and interpretation**

* Prediction aims to make accurate prediction based on the training data using patterns.
  + *To minimize the errors and maximize the accuracy to predict unseen data*.
  + Algorithm with predictive nature
    - Neural network
    - Random forest
    - Gradient Boost
* Interpretation focuses on explaining relationships between input features and predictions.
  + *To explain impact of each feature on outcomes to make the model prediction trustworthy*.
  + Algorithm with interpretable power
    - Linear regression
    - Logistic regression
    - Decision trees.

**Supervised Learning Workflow**

A diagram of a data flow

Description automatically generated

1. **Development-test split**

* Data splitting into development data for training and test data for evaluation.

A diagram of data processing

Description automatically generated

1. **Hyperparameter tuning**

* It regulates model complexity where we split the development data into training and validation data by testing out different subsets of the hyperparameters
* **# of hyperparameters determine the level of model complexity.**

A diagram of a process flow

Description automatically generated

1. **Optimal model training**

* After we select the best set of hyperparameters, we apply those set to train the development data.

A diagram of a training process

Description automatically generated

1. **Model evaluation**

* Evaluate the trained model with the test data to generate the performance metrics.

A diagram of a test model

Description automatically generated

**Data Splitting**

* With the whole dataset to perform supervised learning, we need to have input-output pair where we divide data into development set into training and validation and test set.
  + Development set is used to train model for hyperparameter tuning.
  + Test set is used to evaluate model’s performance on new and unseen data.
* Test set is designed to provide *unbiased evaluation of final model performance* and to indicate *how model will perform when it encounters new data for model evaluation*.
* **Splitting strategies**
  + **Stratified splitting**: it ensures class distribution in both development and test sets is balanced across all splits and mirrors that of original dataset.
  + It is suitable for **highly imbalanced data** in classification tasks where certain class is underrepresented.

A diagram of a diagram of a test

Description automatically generated with medium confidence

**Model hyperparameter**

* Hyperparameter tuning is process to find best hyperparameters for models for performance improvement.
  + **Model parameters** are learned from the data *during the training process*.
  + **Model hyperparameters** regulate the model ability to learn from the data *before training*.
* Hyperparameter regulate model complexity by controlling the degree to fit the curve to the data. As fitting data from worst to best, the degree of model complexity will be less to more complex as well.
* We need to find a “**sweat spot**” to *capture underlying noise and distribution* so that it can *generalize better for unseen data* 🡺 to find the optimal complexity.

A diagram of a normalized curve

Description automatically generated with medium confidence A diagram of a graph

Description automatically generated

* We need to try to find an optimal point to have low bias and low variance.

**Bias-Variance Tradeoff**

* Proper hyperparameter tuning can help find right balance between bias (underfitting) and variance (overfitting) to *ensure model generalize well to unseen data*.
  + ***Overfitting 🡺 high variance low bias***
    - Poor generalization due to complexity
    - Learning noise in training data
    - High complexity, low training error, high testing error.
  + ***Underfitting 🡺 low variance high bias***
    - Poor generalization due to simplicity
    - Not flexible enough to learn concept
    - Low complexity, high training error, high testing error.
* If the model is not sensitive to change of training data, meaning that it captures underlying distribution well, then prediction outcome will not change a lot.

**Ways to prevent overfitting and underfitting**

1. Train-test split:

* Train model with training data
* Keep checking model performance with validation data
* Detect sweet spot when validation performance is decreasing.

1. K-Fold Cross Validation:

* **Data splitting**:
  + In K-fold CV, the dataset is randomly divided into k subsets with equal sizes.
  + For each iteration, model is trained on **K-1 folds as training data** and evaluate on remaining **1-fold as validation data**. The process will be **repeated k-times**, with each fold being used once as validation set.
* **Training and evaluation process:**
  + During each iteration, one of folds as validation and K-1 folds as training will be combined as training set.
  + The model is trained on training set and evaluated on validation set.
* **Average results**:
  + After completing all k iterations, the model’s performance is **averaged over k-folds** to provide prediction of how model is expected to perform on new data.

A graph of a graph

Description automatically generated with medium confidence

1. Grid-search Cross validation:

* It involves an exhaustive search over a manually specified **subset** of hyperparameter space.
* The goal is to find the best combination of hyperparameters that maximize the model performance.

A screenshot of a computer

Description automatically generated

* + The total # of combination in the grid is 3 \* 3 \* 2 = 18.
* **Counting # of times for model training**
  + When we have 2 hyperparameters for distance metrics and 3 hyperparameters for # of neighbors for model, the # of hyperparameters is 2 \* 3 = 6.
  + In the whole process, the grid search CV involve the different subsets of hyperparameters with k-fold CV.

A screenshot of a computer code

Description automatically generated

A screen shot of a computer

Description automatically generated

* + - Since we have 2 distance metrics and 5 neighbors for grid search CV with 3-fold CV, the total # of the model training will be 2 \* 5 \* 3 = 30.
    - Since we need to extract the best model with the optimal best set of hyperparameters, there will be extract 1 time for final model training, and it will be 2 \* 5 \* 3 + 1 = 31 times for total model training to get best score.

**Model Evaluation**

**Metrics for Classification**

* Threshold-based metrics (define threshold to determine point is classified one way or the other)
  + *Classification accuracy*
  + *Precision, recall & F1-score*
* Ranking-based metrics (use score to rank specific range of samples)
  + *AUROC*

**Threshold-based metrics: Confusion matrix**

A diagram of different negatives

Description automatically generated

* **TP**: correctly predicted positive instances.
* **TN**: correctly predicted negative instances.
* **FP**: incorrectly predicted positive instances.
* **FN**: incorrectly predicted negative instances.

**Threshold-based metrics: Accuracy**

A black and white text

Description automatically generated

* It aims to calculate **ratio of *correctly predicted instances*** to ***the total # of instances***.

A diagram of a scale

Description automatically generated with medium confidence

* Limitation of classification accuracy
  + **Misleading in imbalanced dataset**: accuracy will be high by predicting majority class and fails to identify minority class.

**Threshold-based metrics: Precision** 🡺 **when the model predicts positive, how often is it correct?**

A black background with white text

Description automatically generated 🡺 out of observations I predicted positive (TP + FP), how many are truly positive (TP)?

* Precision is **true positive** by **predicted positive**.
* **Precision is linked with predicted data**.
* High precision
  + When the model predicts a positive result, it’s usually correct with few false positives.
* Low precision
  + The model has high rate of false positives, indicating it incorrectly predicts negative cases as positive.
* *Use precision when the model can provide as less negative as possible.*

**Threshold-based metrics: Recall** 🡺 **how well the model capture all positive actual cases?**

A black background with white text

Description automatically generated 🡺 out of truly positive, how many observations did I predict positive?

* **Recall is linked with original data**
* It measures the proportion of actual positive cases the model correctly predicted.
* Recall is **true positive** by **actual positive**.
* *Use recall if model can accurately classify actual positives.*
* **When to use Recall**
  + If the **cost of misprediction or missing a positive (FN) is high**, you should prioritize Recall.
  + Reason: Recall measures the proportion of actual positives correctly identified, reducing the likelihood of missing true positives. This is crucial when false negatives are costly, such as in medical diagnoses or fraud detection.

**Threshold for precision and recall**

* Set a threshold to decide if the prediction belongs to positive class or the other.
* Probability of 50% to segment 2 different classes.
  + Lower threshold 🡺 increase recall and decrease precision
  + Higher threshold 🡺 increase precision and decrease recall.

**Threshold-based metrics: F-1 score**

A black and white sign with white text

Description automatically generated

* It is suitable for **imbalanced dataset** by measures harmonic means of precision and recall, balancing 2 metrics by **penalizing extreme values**.

**Rank-based metrics: Receiver Operating Curve (ROC)**

* The ideal ROC moves sharply up to the top-left corner, meaning high true positive rates with low false positive rates.

A graph with a line

Description automatically generated

* **AUROC**
  + The goal is to find the largest area under ROC to ensure model can provide optimal performance for classification.

**Metrics for Regression: R-sq**

* R-Sq measures **proportion of variance in the DV explained by IV**.
* It ranges from 0 to 1, where a higher value indicates a better fit.

**Metrics for Regression: Adjusted R-sq**

* **R^2**: it represents the ***proportion of variance in the target variable explained by the features.***
  + Adding explanatory features to a model always increases R^2, even if the features are not relevant.
* **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**.

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**

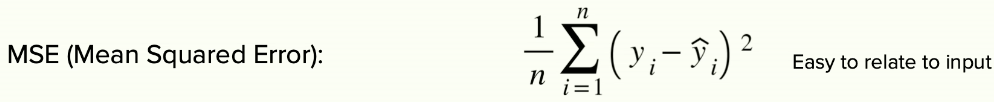
A close up of text

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.

**Metrics for Regression: Mean Squared Error (MSE):**

* It calculates the average squared difference between predicted and actual values.
* It aims to penalizes large errors more heavily, making it sensitive to outliers.



**Data Preprocessing: Imbalanced Dataset**

Imbalanced class occurs when ***one class significantly outnumbers another*** in a classification tasks. It will lead to biased model and misleading accuracy.

**Stratified sampling**:

* to ensure each class is represented equally during train-test split 🡺 to preserve distribution of majority and minority class in the data splitting.

**Random Undersampling of majority class**

* to reduce the # of sample in the majority class to equalize the class distribution without changing minority class 🡺 ***to correctly predict samples from minority class.***
  + Identify majority and minority classes
  + **Randomly remove samples from majority class** until the size is balanced.

**Random Oversampling of minority class**

* To replicate existing data points in minority class without changing the majority class 🡺 ***to reduce bias to the majority class and correctly predict majority samples***.
  + Identify minority class
  + **Duplicate minority class by randomly** selecting samples from minority class and duplicate them until minority class size matches that of majority class.

**SMOTE (Synthetic Minority Oversampling Technique)**

* ***To create synthetic samples for minority class***, rather than simply duplicating existing samples.
* By generating synthetic data, **SMOTE increases # of samples in the minority class, leading more balanced dataset without changing majority class**.
  + For each minority class, find its nearest neighbors.
  + Create synthetic samples by joining between 2 chosen samples and a randomly selected neighbor.

**ADASYN (Adaptive Synthetic Sampling)**

* To balance classes by focusing on **hard-to-learn samples near decision boundaries**.
  + It creates ***more synthetic samples where classification is difficult*** by making adaptive oversampling in minority samples.

**Data Cleaning**

**Duplicated data**

1. We need to drop the duplicated data when the data is unique by unique identifier to indicate unique and same value elsewhere.

* Find # of duplicated or unique values using ***.duplicated()***.

A screenshot of a computer

Description automatically generated

* Show the duplicated rows using ***.duplicated (keep = False).***

A screenshot of a computer

Description automatically generated

* Remove all other duplicated rows and only keep 1st row using ***.duplicated (keep = ‘first’)***.

A screenshot of a computer

Description automatically generated

* Remove duplicated columns subset while keeping all other distinct columns

A screenshot of a computer

Description automatically generated

* + ***Data.loc[Data.duplicated(subset = [‘col’], keep = ‘first’)]***
    - We need to remove the duplicated column in ‘purchase\_id’ while keeping other columns intact.
* Drop all duplicates in the dataset using ***.drop\_duplicates().***

A screenshot of a computer

Description automatically generated

* + inplace = False will return a new dataframe without changing the original dataframe.
  + inplace = True will change the original dataframe.
  + keep = False will drop all duplicates.
  + keep = 'first' will keep the first occurrence of each duplicate.
  + keep = 'last' will keep the last occurrence of each duplicate.
  + subset = None will return and consider all columns in the dataset.

**Missing Data**

1. **ways to check missing data.**

* Check missing value information using ***.isna ()*** or ***.isnull ()*** to have binary value.

A screenshot of a computer

Description automatically generated

* Check the # of the missing values in each column using ***.isna ().sum ()***

A screenshot of a phone

Description automatically generated

* Check the total # of the missing value in the dataset using ***.isna().sum().sum()***

A close-up of a computer code

Description automatically generated

1. **Ways to manipulate missing values**

* **Drop rows for missing values with all columns** using ***.dropna()***

A screenshot of a computer

Description automatically generated

* + If observation contains at least 1 missing value (NaN), this row along with other info will be dropped as well.
* **Drop rows for missing values with particular columns** using ***.dropna(subset = [‘col’])***

A screen shot of a computer

Description automatically generated

* + We can enable to specify the particular column by dropping the missing values.
* **Drop rows with all missing values** using ***.dropna(how = ‘all’)***

A screen shot of a computer

Description automatically generated

* Fill the missing value with **constant** using ***.fillna(constant)***

A screenshot of a computer

Description automatically generated 🡺 A screenshot of a computer

Description automatically generated

* Fill the missing value with **continuous statistics** using ***.fillna(df.mean())***

A screen shot of a computer code

Description automatically generated

* Fill the missing value with **categorical statistics** using ***.fillna(df.mode())***

A screenshot of a computer

Description automatically generated

* Fill the missing value with **adjacent value** using ***.fillna(method = ‘ffill’)***

A screen shot of a computer code

Description automatically generated 🡺 use value above.

1. **Simple imputer** for missing values using ***SimpleImputer()***

* Replace the missing values with ***mean, median and most\_*frequent** values of that column.

A screenshot of a computer code

Description automatically generated

1. Create a **dummy column** while filling the missing value

A screenshot of a computer program

Description automatically generated

* Creating dummy column can indicate which specific value is missing or not to **reveal the meaningful pattern in the data.**
* The dummy column can indicate the detect the relationship between missingness and target variable.
* It can detect the bias caused by simply filling the data by the mean value without considering that data is missing initially.

**Feature Rescaling**

* Rescaling is key to ***model training and distance comparison*** with the same scale.
* **Z-Score standardization** using ***StandardScaler()***
  + ***to scale the data to mean of 0 and SD of 1***.



* + ***StandardScaler.fit\_transform(df.[[‘col1’, ‘col2’]])***

A screenshot of a computer code

Description automatically generated

* **Min-Max Rescaling** using ***MinMaxScaler()***
  + ***To scale the features to a fix range [0, 1]***
  + Map minimum value to 0
  + Map maximum value to 1

A black text on a white background

Description automatically generated

* + The goal is to remove all negative values in the original data.

A screenshot of a computer code

Description automatically generated

**Dealing with Skewness**

* Skewness will affect data sensitive to outliers

A graph of a number of columns

Description automatically generated with medium confidence

* Normalize dataset using ***square-root transformation***

A graph of a graph

Description automatically generated

* Delivering normal transformation using ***log-transformation***

A graph of a graph

Description automatically generated

**Detecting and Removing Outliers**

* Outlier plays the similar role as missing data which doesn’t contribute too much meaning in the whole dataset.
* Use **boxplot and bar plot** for outlier detection

A screenshot of a graph

Description automatically generated

* **Outlier reduction**
  + Drop outlier data
  + Treat it as missing value
  + Encode with dummy variable.

**Feature Engineering**

**Binning 🡺 *pd.cut( )***

* To divide continuous variable into discrete intervals (bins) for transforming continuous data into categorical data.
  + Binning can reduce the impact from outliers by grouping data into broader categories.
  + Binning can help detect and represent non-linear relationship between feature and target.
* We initially **decide the interval** under the column based on continuous variable and apply binning **by splitting the column into bins** with **defined labels** for each interval. Therefore, each value will be **granted with categories defined by binning**.

A screenshot of a table

Description automatically generated

***One-hot Encoding 🡺 pd.get\_dummies () or sklearn.preprocessing.OneHotEncoder***

* To convert categorical variables by encoding transforms categories into binary features 🡺 allowing models to process categorical data effectively.
* Create numerical variable from categorical variables by marking presence as 1 and 0 otherwise.
* For the categorical feature with **N unique categories**, one-hot encoding creates N new binary columns with the prefix for each unique categories and *assign 1 to the column corresponding to the category of the rows, and 0 otherewise*.

A screenshot of a black screen

Description automatically generated 🡺 A screenshot of a computer

Description automatically generated

**Join the data**

**Method 1: Join datasets by set index using *df.join ()***

* **Left Join: *A.join(B) | A.join(B, how = ‘left’)***

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.
* **Right join: *A.join(B, how = ‘right’)***

A screenshot of a computer

Description automatically generated **🡺** A diagram of a diagram

Description automatically generated with medium confidence

* **Inner Join: *A.join(B, how = ‘inner’)***

A screenshot of a computer

Description automatically generated **🡺** A blue circle with black text

Description automatically generated

* **Outer join: *A.join(B, how = ‘outer’)***

A screenshot of a computer

Description automatically generated **🡺** A diagram of a diagram

Description automatically generated with medium confidence

**Method 2: Join datasets by columns using *pd.merge(A, B)***

* **Inner join: *pd:merge(A, B, on = ‘common\_col’, how = ‘inner’)***

A screenshot of a computer code

Description automatically generated

**Natural Language Processing**

1. **Token (smallest unit)**

* a single work or part of word
* ["I", "love", “NLP”, “.”] or [“I”, “l”, “o”, “v”, “e”, “,” …]

1. **String (sequence of characters)**

* A list of characters representing raw text.
* "I love NLP." Or ‘I love’

1. **Document (unit of cohesive text)**

* A cohesive unit of text that contains one or more strings
* “I love NLP." Or ‘I love’

1. **Corpus (collection of documents)**

* Collection of multiple documents
* ["I love NLP.", "NLP is fun and interesting!"]

1. **Vocabulary (largest representation)**

* A set of unique tokens found in entire corpus
* Unique words, punctuation, and case-sensitive
* {"I", "love", "NLP", ".", "is", "fun", "and", "interesting", “!”}

**Tokenization**

* A process of breaking a document into a list of smaller units (tokens) to prepare text for further analysis by converting unstructured data into structured data.
* **To convert each token into feature vector for later model performance** after we have tokenized raw data into a simpler form.
  + **Token**: an **individual element resulting from tokenization**, including words, punctuation, or subwords.
  + **Vocabulary**: the set of **unique tokens** in a corpus, representing complete list of words across all documents.

**Bag of Words (BoW)**

* To convert a document or corpus into a collection of tokens, *ignoring grammar and word order,* while **keeping track of frequency of each word**.
  + Word order ignored: ignore the sequence of document.
  + Frequency-based representation: the model counts for occurrence of each word in the document.
  + Vocabulary: the unique words across entire corpus where size of vocabs decides the dimensions of the BoW representation.

**n-Grams**

n-Grams is **contiguous sequence of n tokens** from a given text

1. **Unigram**: consists of single tokens

* "I love NLP" → ["I", "love", "NLP"]

1. **Bigram**: consist of 2 consecutive tokens

* "I love NLP" → ["I love", "love NLP"]

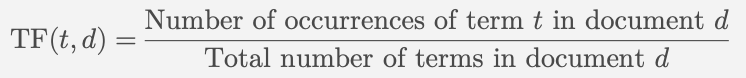
1. **Trigram**: consists of 3 consecutive tokens

* "I love NLP" → ["I love NLP"]

Every time it moves to the next gram, it maintains the same size of n-gram but move by 1 step until the token reaches to the end.

**Term Frequency (TF)**

* **The # of times a specific term is seen per document** 🡺 # of times term t appears in a document d in the corpus.



* Common words will indicate high frequency, even if they provide little meaning.

A screenshot of a cell phone

Description automatically generated

**Document Frequency (DF)**

* **The # of different documents containing the same term in the corpus.**



* Once we know how those terms without significant meaning, we can use DF to see how often they appear in each document.

A white background with black text

Description automatically generated

**Stopwords**

* Stopwords are common words that carry little meaning in the text.
* We tend to remove the stopwords since it will lead high DF while containing no unique meaning and little understanding of context.

**TF-IDF (Term Frequency-Inverse Document Frequency)**

* It aims to evaluate importance of a term in a document relative to corpus to identify terms by balancing TF and DF.
* The goal is to **downweigh common terms** that appear across many documents **to reduce their influence on the final text representation**.
  + **Problem with TF**: common words like “the,” “is,” and “and” appear frequently but don’t provide much information.
  + Solution: use Inverse DF (IDF) to reduce weight of frequent terms and highlight important ones.

1. **Term Frequency (TF):**
   * How often a term t appears in document d.
2. **Inverse Document Frequency (IDF):**
   * A measure that *decreases the importance of terms appearing in many documents*.

**Topic Modeling**

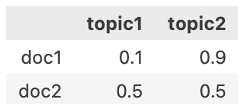
An unsupervised learning framework used to **discover hidden topics in the corpus** with assumption:

* ***Documents are composed of topics.***
* ***Topics are collections of related terms.***
* Document 🡸 topics 🡸 terms
  + **Topic**: **a collection of terms that frequently appear together** that defines a topic.
  + **Per Topic Term Distribution**: what terms make up each topic?

A screenshot of a game

Description automatically generated

* + **Per Document Topic Distribution**: what topics make up each document?



**Latent Dirichlet Allocation (LDA)**

* Under the assumption from topic modeling, LDA believes that **distribution in topic modeling is determined based on TF patterns**.
* 2 metrics in LDA:

1. **Phi: Per Topic Term Distribution**
   * Phi indicates *how each term distributed in each topic*.
2. **Theta: Per Document Topic Distribution**
   * Theta suggests *how each topic distributed in each document*.

* LDA aims to predict topics of documents based on their content by identifying latent topics in the corpus.

**Supervised Machine Learning Models**

**Regression Model**

**Simple Linear Regression Model**

* A model contains DV and IV to indicate the correlation between 2 variables with considering coefficient, intercept, and error.

A screenshot of a computer

Description automatically generated

* The process of linear regression involves fitting the line based on the existing data as close to the observation as possible.

**Ordinary Least Squares (OLS)**

* There always is gap between observed data and predicted values from regression line. OLS aims to minimize the gap (SSE) by fitting the line that minimize the total squared difference between observed value (y) and predicted value (y^).
* Since minimizing SSE requires quadratic closed form, we can calculate W0 and W1 directly *without iterative methods*.

A math equations on a white background

Description automatically generated

* **Coefficient Interpretation of OLS**
  + **On average, the amount of change in the dependent variable (DV) for a one-unit increase in the independent variable (IV), holding all other variables constant.**

**Gradient Descent**

* An iterative method to minimize SSE in regression by adjusting model parameters W0 and W1 to reduce error.
  + The gradient is a vector of partial derivatives that points in **direction of the steepest increase or decrease.**
* The goal of Gradient Descent is to minimize error by taking steps in the opposite direction of gradient, where the minimization problem follows the **opposite direction** of the gradient to decrease the objective function 🡺 **minimize SSE**
  + **Global max/min**: the absolute best solution over the entire dataset.
  + **Local max/min**: the best solution within specific neighborhood.

**Multiple Linear Regression**

* Multiple linear regression model extends simple linear regression by modeling the relationship between a DV and 2 or more IVs.

A white background with black text

Description automatically generated

* **Adding constant:** 
  + Adding a term in regression plays as a baseline level for DV in case all coefficients for IVs are set to 0.
  + It can enable the better fit the data by adjusting the baseline for the DV.
  + It can avoid the bias term **by preventing the model from being forced to pass through the bias term.**
* **Multi-collinearity:**
  + When two or more predictor variables in the regression model are highly correlated with each other. In other words, one predictor can be linearly predicted from the others with a significant degree of accuracy
    - Since the data we collected could potentially have strong collinearity, one feature can be expressed as a **linear combination** of the others.
  + Since we need to make sure trained model should be robust to slight variation of data, we need to ensure the data noise can be cleaned as much as possible.
    - We can *change the data a little bit and see how the coefficients change in what extent with respect to different features*

**Regularization**

Goal: to address ***multicollinearity*** from linear regression by penalizing the extreme weight or coefficient for certain features

**Regularization: Ridge Regression (L2 norm)**

* Add an L2 regularization penalty to the standard linear model to address issues of relevant features and prevent overfitting.
  + When predictors are highly correlated, ridge regression stabilizes the coefficient estimates by reducing the variance of model.
  + **Prevent overfitting**: by introducing penalty term, ridge regression reduce overfitting by keeping all coefficient relatively small and uninformative feature to be smaller.

**Regularization: Lasso Regression (L1 norm)**

* Aim to perform ***feature selection*** by shrinking some of regression coefficient to **exactly 0**, simplifying model and addressing overfitting.
* **L1-norm**: forces absolute values of coefficients to be small to 0.
  + **Shrinkage**: Lasso can shrink magnitude of coefficient and drive some coefficient to 0.
  + **Feature selection**: for coefficient remained non-zero, they can be considered as informative feature that contribute to the prediction.

**Regularization: Elastic-Net Regression (L1 and L2 norm)**

* It uses both L1 and L2 norm which can eliminate inactive features.
* Balance L1 and L2:
  + L1: controls the Lasso to shrink some coefficient to exactly 0 for feature selection 🡺 to remove some informative feature entirely.
  + L2: controls the Ridge to shrinking coefficient without setting them to 0 to reduce multicollinearity 🡺 to remain correlated predictors in the model

A white background with black text

Description automatically generated

**Regression / Classification Model**

**Logistic Regression (linear models for classification)**

* Then prediction is given with **binary labels (-1 and 1),** we can use logistic regression to apply for binary classification tasks.
  + Add a threshold: we can treat the output of logistic regression function.
  + Setting threshold at 50% to get class prediction, where any prediction above or below the threshold will be considered as either class.

A graph of a graph with red and blue dots

Description automatically generated

**K-Nearest Neighbors Model**

* k-NN is a simple baseline model because it is intuitive, easy to implement, and non-parametric 🡺 **NO estimation of parameter at all**.

A close-up of words

Description automatically generated

* Steps

1. the training data is used to make prediction.
2. Given the training data, we predict the label of new data points by projecting them into the same space as the training data.
3. It estimates distances between this new data and all other training data by Euclidean Distance.
4. Nearest neighbors are identified based on calculated distances. The parameter ‘k’ determines the # of neighbors affects the prediction.
   1. **Classification**: it uses **majority voting** among the nearest neighbors to decide the label for the new, unseen data.
   2. **Regression**: it **averages the values of nearest neighbors**.

* **kNN property**
  + **No need to train the model**: kNN directly uses ***training data for prediction***.
  + Requires extensive prediction effort since each new, unseen data need to compute distance to all points in the training.
  + **Curse of dimensionality**: kNN cannot only perform in high-dimension space that leads to distorted performance.

**Decision Tree Model**

* Decision tree can either perform classification for categorical outcome or regression for continuous outcomes.
* Rule of tree
  + When putting an input data from root, it will end up with the particular leaf as terminal, ensuring every prediction from input with an outcome.
* Metrics for leave node
  + **Classification**: tree optimal splits based on Gini Index or Entropy and use Information Gain to determine optimal split. 🡺 **less Gini, higher IG.**
  + **Regression**: use variance or MSE to measure impurity and SSE to determine optimal split.
* Metrics for nodes
  + **Sample means**: it indicates average # of % predicting input data will fall to this leaf node.
  + **Class**: it indicates label predicting label fall into.
  + Once the input data is entered to the root, we will be informed of *what proportion of that input will fall under which particular category*.
* **Goal of decision tree**
  + To make the optimal split at each node, aiming to **reduce impurity in the most immediate and effective way in each step** to ensure all sample can correctly fall into particular leaf only belong to one class with maximum probability.
    - The impurity is gradually decreased as tree goes down.
    - **Overfitting concern**: the tree is built to **have complete purity making # of leaves = # of samples with 100% accuracy and 0 training error**.
      * *Imposing hyperparameters (max\_depth) to enable tree to have enough predictive power while maintaining optimal depth of tree*.

A diagram of trees with black text

Description automatically generated

* Limitations
  + Individual tree is **highly unstable** for small change in training data can lead to large change in tree structure.
  + Poor performance on continuous outcome for **limited model capacity** with fixed # of leaf nodes under one tree.

**Ensemble Method**

* A technique to combine multiple decision tree to improve overall model’s predictive performance.

A diagram of a data analysis

Description automatically generated

* **“Wisdom of Crowd”**: ensemble methods help to address decision tree limitations by combining multiple trees to achieve better performance with collection of learners.
  + Ensemble method aims ***to reduce model variance*** by combining multiple trees to improve predictive power 🡺 best when each tree is not correlated.

1. **Bagging (Bootstrap Aggregation)**
   * Create separated training samples by **sampling data with replacement.**

A diagram of data analysis

Description automatically generated

* + Each training sample is used to train a separate decision tree.
    - **Regression tree:** average output of all individual trees.
    - **Classification tree:** majority voting.

1. **Random Forest Model**
   * A specific type of bagging with random feature selection.
     + **Bagging**: By sampling with replacement, each dataset (bootstrap sample) may contain *duplicated data and some potential excluded data*.
     + **Random feature selection**: *randomly selecting subset of features at each node when splitting each node* 🡺 prevent model from relying too heavily on most dominant features, **making trees more decorrelated to each other**.
   * Each of learner with different bootstrapped samples and subset of features is a decision tree, where each decision tree is **high variance** and enabling all other feature with **less dominant power have a say in the final prediction**.
   * **Hyperparameter:**
     + n\_estimator: # of trees in the forest
     + max\_features: # of features to splits at each node.
   * **Feature importance at Random Forest**
     + **Impurity reduction** at each split:
       - In each decision tree, use Gini, entropy, or variance to quantify impurity reduction of each split.
       - The importance of each feature is measured by summing IGs that feature provides across all trees and all nodes.
     + Feature importance is calculated as the **decrease in node impurity weighted by probability of reaching that node** 🡺 higher the value, the more important the feature.
2. **Gradient Boosting Tree Model**
   * It aims to **sequentially** train the series of weak learners to correct the errors of their predecessors.
     + Each new tree is ***trained on the residuals / errors from the previous tree’s predictions* by adding more weights for misclassification**.
     + the process helps model learn from its mistakes, focusing on more **hard-to-classify cases** in each iteration.
     + The boosting process uses **gradient descent** to reduce error by moving in direction that minimizing the loss function.
3. **Model Staking Technique**
   * Stacking is used to combine models to improve the predictive performance, involving training metal-model by implementing a base model as input features.
     + The meta-model (final estimator) is trained to predict target variable using prediction of base models (estimators)
     + Be mindful that all base models should focus on either classification or regression.

A screen shot of a computer code

Description automatically generated

**Perceptron and Multilayer Perceptron**

* **Perceptron**: A simple linear classifier that computes a weighted sum of inputs and applies a step function to make binary predictions.
* **Multilayer Perceptron (MLP):** A neural network with multiple layers that can model complex, non-linear relationships using activation functions and backpropagation.

**Multiclass, Multilabel classification model**

1. **Multi-class Classification**: more than 2 categories or classes as target variables
   * Models that can perform multi-class or multi-label classification
     + Logistic regression with multi\_class = ‘multinomial’
     + KNeighbors Classifier
     + Decision Tree Classifier
     + Random Forest Classifier.
2. **Multi-label Classification**: more than 2 labels to instances to predict multiple binary labels for each instance.
3. **One-vs-Rest multi-class classification**
   * Multi-class problem is broken into multiple binary classification problems
     + One class belong to that class, and the rest of all belongs to the other class.
   * Training:
     + A binary classifier is trained to distinguish between each class and all other class combined in **imbalanced data** 🡺 if there are K classes, the algo creates K binary classifiers.
   * Prediction:
     + Each K binary classifier makes a prediction, providing a **confidence score** for input belonging to their respective class 🡺 **class with highest confidence score is selected as predicted class**.

A close-up of a paper

Description automatically generated

**Dimensionality Reduction: Feature selection**

**Motivation of Feature Selection**

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

**Linear Regression with Lasso Regularization (L1)**

* It turns uninformative feature coefficients to exactly 0 and select the features.

**Random Forest with Tree-based model.**

* The effort of impurity reduction will be ranked as feature importance.
* The more impurity reduction in each split, the more important of that feature will be.

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 interaction.
  + Perform statistical test on each feature independent of all others.
  + To identify the features with the strongest individual relationship to the target.
* **Select best features regarding to the target feature independently.**
  + Conduct F-test for each training features against target variable.
    - F-test estimates the degree of linear dependency between feature X and target y.
  + Keep the most important n features.

**Recursive Feature Elimination (RFE)**

* To iteratively **remove** the least important features, reducing the feature step by step, and evaluate the model’s performance at each step.
  + Train the model to assigns importance to features.
  + **Rank** features based on their importance *by coefficient or feature importance* score.
  + Eliminate least important features **iteratively**.
  + Repeat the process of elimination until the desired # of feature is reached.

A screenshot of a computer

Description automatically generated

**Dimensional Reduction**

**Feature Extraction**

Motivation: Feature extraction involves transforming original features into a new feature space that captures most relevant information from the data, reducing redundant features, improve model performance, and help with visualization.

* Create a new feature space by combining or transforming existing features.
* Unsupervised methods 🡺 **Maintain as much information or variance as possible**
* Supervised methods 🡺 maximize separation between classes.

**PCA (unsupervised framework)**

* **Motivation of PCA**
  + Instead of applying PCA to all 1000 features, we can focus on a subset of features for dimension reduction.
  + Use PCA to verify if the subset’s PCs can **preserve majority of variance** from original features.
  + **Variance explained by PCs**
    - Each PC explains the portion of the total variance in the data.
    - Cumulative variance explained by PCs, we can determine how much information is preserved.
  + 90% of variance threshold: if the PCs from subset can explain >90% variance, the subset is adequate.
* It aims to transform data into a new feature space by focusing on directions of principal components that capture the most variance.
  + Variance as Information:
    - PCA assumes **direction** of high variance in the data are the most informative.
    - Find PCs by identifying directions within **linear combination of features** **that explain the most variance**.
      * 1st component: the direction that explains the maximum variance in the data.
      * 2nd component: a direction perpendicular to the 1st PC, capturing the next highest variance.
      * Subsequent components: each new component is orthogonal to all previous PCs with remaining variance.
  + **Combining collinear Features**
    - Features highly correlated (collinear) will be combined into a single PC, therefore, reducing redundancy.
  + Projection:
    - Data is projected onto a subspace with fewer dimensions while preserving as much variance as possible.
* **Process**

1. **Centering the data**
   * Subtract the mean of each feature to ensure data is centered around 0 to align PCA with direction of maximum variance.
2. **Find PCs**
   * Identify direction (linear combination of features) that explain the most variance.
     + Direction of linear combination of features
     + Explain maximum variance
3. **Repeat until maximum components**
   * # of components cannot exceed # of original features.
4. **Projection.**
   * Transform data by projecting it onto PCs.

A screenshot of a graph

Description automatically generated

* PCA in Explained Variance
  + Visualize how each PC explains the variance of that feature, where each PC may contain the combination of features that share the correlations.
  + The longer vector, the higher variance of that PC.
  + Tell how each PC diminish each explained variance, indicating the each following PC will explain less than previous ones.

A screen shot of a graph

Description automatically generated

**Unsupervised Learning Framework**

Clustering aims to group data based on their features without using predefined labels or targets.

* Clustering groups data into K clusters based on feature similarity.

**K-Means Clustering**

* Partition data into k clusters by minimizing distance from each point to its nearest cluster center, where each data belongs to cluster with the nearest means (**centroid**).
  + K-Means uses **Euclidean Distance** to measure *similarity between points and centroids*.
* **Mechanism**:
  + **Initialization**: Choose k random points as initial cluster centroid.
  + Assignment step (A)
    - For each data, assign it to the nearest centroid based on Euclidean Distance.
    - Fix means 🡺 assign all datapoints to their closest mean.
  + Update step (B)
    - Recalculate the centroid by taking mean of all points assigned to each cluster.
    - Fix cluster assignment 🡺 recalculate means.
  + Repeat and alternate between A and B until
    - Convergence: centroids stop changing
    - Maximum iteration: a limit is reached.
* **Evaluation of K-Means using within cluster sum of squared distances (SSD)**
  + It measures how well K-Means algo has grouped the data into clusters.
  + **High SSD**
    - Points are **far from** their assigned cluster centers.
    - Clusters are spread out and poorly compact.
  + **Low SSD**
    - Points are **close to** their assigned cluster centers
    - Clusters are tight and well-formed.

**Hierarchical Agglomerative Clustering (HAC)**

* HAC is a **bottom-up clustering algorithm** where each data starts from its own cluster, and clusters are merged iteratively based on closeness until a single cluster remains. This process creates a binary tree called dendrogram, which represents hierarchy of clusters.

A graph of a number of individuals

Description automatically generated with medium confidence

* Process

1. Initialization: Every point starts as its own cluster.
2. Step A: to **find the closest pair of clustering** using distance metric and linkage criteria.
3. Step B: to **merge the closest clusters into one**.
4. Go back to A until there is one cluster left or until a stopping condition is met.

* Metrics to define closeness: **Linkage**
  + Single Linkage (Minimum Distance)
    - Shortest distance from item of one cluster to item of the other.
  + Complete Linkage (Maximum Distance)
    - Longest distance from item of one cluster to item of the other.
  + Average Linkage
    - Average distance of item in one cluster to items in the other.
  + Ward’s Method
    - Merge clusters with smallest increase in total variance.
    - Only works in Euclidean distance.

**Recommendation Engines**

**Types of Recommendation Engines**

1. **Content-Based Filtering**

* Recommends items similar to what user himself has already liked based on item features.
  + “If you liked item A, you will like similar items.”
* Steps:
  + Extract features of liked items.
  + Compare items to user preferences using similarity metrics (Euclidean Distance)
* Calculate pairwise distance with Euclidean Distance between every item where each value represents distance between 2 data point, indicating **how similar of different they are** 🡺 the larger number, the less similar between items
* Same units for matrix axis on X and Y and all 0 in diagonal

A screenshot of a computer

Description automatically generated

* + Distance between house 0 to house 1: 3.74
  + Distance between house 0 to house 3: 1.12
  + Distance between house 2 to house 3: 2.61

1. **Collaborative Filtering**

* Recommends items by finding similar users and suggesting what they liked.
* If 2 users have similar tastes, it assumes that preferences can be shared.
  + “People like you also liked those items”
* Approaches
  + **User-User Collaborative Filtering** 🡺 find users similar to active user and recommend items they liked.
  + **Item-Item Collaborative Filtering** 🡺 find items similar to what the other active user has rated/liked and recommend them.
* **Process** 
  + Build a User-item Matrix that tracks interactions

A screenshot of a computer screen

Description automatically generated

* + Use Cosine Similarity to **measure user similarity**

A screenshot of a computer

Description automatically generated

* + Find top **similarity users** to the target user.

A screenshot of a computer code

Description automatically generated

* + Recommend items liked by these similar users but not yet seen by the target user.

A screenshot of a computer

Description automatically generated

* + - MapReduce (0.57): User 9’s high similarity led to this recommendation.
    - Postgres & MongoDB (0.51): Users with similar data-tech interests influenced these.

**Issues on Collaborative Filtering**

1. **Cold Start Problem:** 
   1. This occurs when new users or new items enter the system. Since there is **no prior interaction history**, CF struggles to provide accurate recommendations.
2. **Sparsity:** 
   1. Recommendation systems often have large interaction matrices **with many missing values**, leading to difficulties in generating recommendations.

**Model Pipeline**

Building model pipeline can simplify and automate process of building a ML workflow, combining data preprocessing and the final model training and prediction into the streamline process.

* Construct **sequential chain of transformers** followed by an estimator to ensure entire workflow is applied consistently and efficiently.
  + **Transformer**: apply ***.fit\_transform()*** during training and ***.transform()*** during testing 🡺 apply only ***transform() in testing*** can prevent **data leakage**.
    - **Preprocessors like scaling, encoding, or feature extraction.**
  + **Estimator**: apply ***.fit()*** during training and ***.predict()*** during testing.
    - **Classifier or Regressor.**

A diagram of a step-by-step process

Description automatically generated

**Components of a pipeline**

1. **Preprocessing:** 
   * implement ***.fit()*** and ***.transform()*** methods during scaling, encoding, or imputing missing values.
2. **Estimator:**
   * The final step of pipeline is the ML model with ***.fit()*** and ***.predict()*** methods

A close-up of words

Description automatically generated

1. **Hyperparameter tuning with Grid Search**

* Use ***“\_ \_”*** for each specific model for hyperparameter tuning.
* Apply pipeline with dictionary of hyperparameters using .fit(X\_train, y\_train)

A screenshot of a computer code

Description automatically generated

1. Pipeline in display

* The **order** of each of parameter inside the pipeline matters.

A screenshot of a computer

Description automatically generated

**Column Transformer**

* To enable to apply different transformation to different subsets of columns, which is particularly useful when dealing with datasets that have mixed datatypes.

A table with numbers and letters

Description automatically generated

* **Column transformers with pipeline**
  + Selective transformation by applying specific preprocessing steps to specific columns.
    - **Transform numerical feature**

A screen shot of a computer code

Description automatically generated

* + - * Fill missing value for column age and fare with median by **SimpleImputer**.
      * Standardize both columns with standard scaler.
    - **Transform categorical feature**

A close-up of a computer code

Description automatically generated

* + - * Use **simple imputing and one-hot encoding** for categorical feature.
  + **Integration with Pipeline with ColumnTransformer**

A computer code with black text

Description automatically generated

* + Visualize pipeline

A screenshot of a computer

Description automatically generated