Fit Prediction for Rent the Runway Data

Gayatri Kharche
A69027270

Wenyi Zhang
A69027271

Esha Sali
A69027358

ABSTRACT

This research focuses on building a personalized recommendation system for Rent the Runway, utilizing machine learning techniques to enhance customer experience and satisfaction. The study leverages a diverse dataset containing user reviews, demographics, rental history, physical attributes, and rental contexts to predict the suitability of rented clothing items. Data preprocessing includes feature standardization, encoding, and handling missing values to prepare the data for modeling. Exploratory Data Analysis uncovers relationships between user attributes, rental preferences, and fit feedback, offering valuable insights for tailoring recommendations. The study implements and evaluates multiple algorithms, including logistic regression, support vector machines, XGBoost, k-nearest neighbors, and advanced deep learning models like convolutional neural networks (CNNs), TabNet and BPR. Metrics such as accuracy, precision, recall, and the F2-score are used to assess model performance, with a focus on optimizing recall to reduce false negatives and improve user satisfaction. Neural networks demonstrate superior performance, particularly in predicting 'fit feedback,' enabling precise recommendations for size, style, and fit. By integrating user attributes with rental contexts, this research provides a scalable and data-driven approach to personalization in fashion e-commerce, significantly enhancing the value proposition for customers and businesses alike.

1 INTRODUCTION

This report focuses on building a personalized recommendation system for Rent the Runway using machine learning techniques. The goal is to design a model that takes into account user demographics, rental history, and physical attributes to recommend clothing that best suits the individual's preferences and needs. By leveraging user reviews, body type information, and rental context (e.g., event type), we aim to create a system that enhances the user experience and drives greater engagement with the platform.

To achieve this, we first conduct an exploratory data analysis (EDA) to uncover patterns and correlations within the dataset, followed by the design and implementation of various recommendation algorithms to personalize suggestions for users.

1.1 Dataset

The dataset used [6] for this analysis comes from Rent the Runway and contains a comprehensive set of user reviews and rental information. It includes multiple features, such as user demographics, details of the rented items, review text, and numerical measurements of physical attributes like weight, height, and body type. Each entry in the dataset is associated with a unique user id and item id, linking the review to a specific user and the item they rented. The dataset also contains ratings on a scale from 2 to 10, providing insights into customer satisfaction, and additional attributes like bust size, weight, height, and body type. The rental context, including whether the item was rented for a vacation, wedding, or other special events, is also included, along with the date of the review and the size of the rented item.

	RenttheRunway
Number of Transactions	135498.000000
Numober of Customers (Unique)	71379.000000
Number of Products (Unique)	5668.000000
Avg Transactions per Customer	1.898289
Avg Transactions per Product	23.905787
Median Transactions per Customer	1.000000
Median Transactions per Product	10.000000
Fraction Fit	0.736158
Fraction Small	0.133389
Fraction Large	0.130452
Number of Active Customers (>5 Transactions)	3005.000000
Number of Active Products (>5 Transactions)	3781.000000
Avg Unique Users per Product	23.873853

Figure 1: Table - Basic Statistcs (After Preprocessing)

1.2 Exploratory Data Analysis

The dataset initially contained 192,544 entries. After handling missing values and data cleaning, the dataset was reduced to 135498 entries, ensuring its readiness

for analysis. Key statistical features were computed for several numerical columns, such as rating, age, and size. For instance, the ratings had a mean of 9.08 with a standard deviation of 1.43, with values ranging from 2 to 10. The age column had an average of 34 years and a standard deviation of 8.11 years, with an age range from 0 years to 117 years. Notably, the age column contained some data quality issues, as 0 years likely represents missing or incorrect data. The size column had an average of 11.11, with values ranging from 0 to 58, where the extremes of size 0 and size 58 could be considered outliers.

During the cleaning process, missing values were handled, and the dataset was checked for any remaining null values. Following these steps, all columns were free from missing data, ensuring a clean dataset for further analysis.

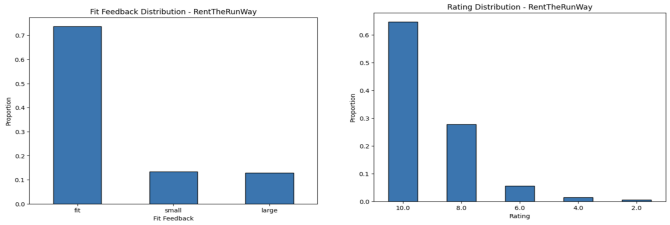

Figure 2: Plots - Basic Statistcs

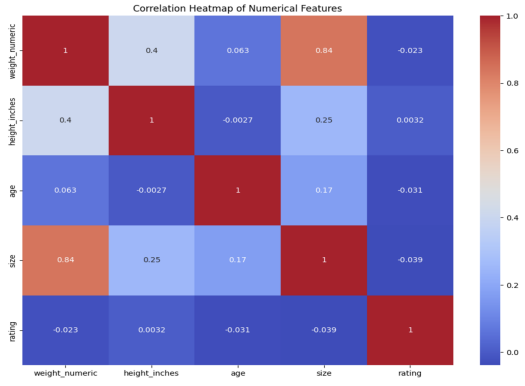

Figure 3: Plots - Correlation

During the exploratory analysis, several key patterns were observed. Figure 2 illustrates that most users report the fit as accurate ("fit"), with a smaller proportion indicating "small" or "large," suggesting that the sizing system is generally reliable. However, the presence of "small" and "large" feedback points to potential areas

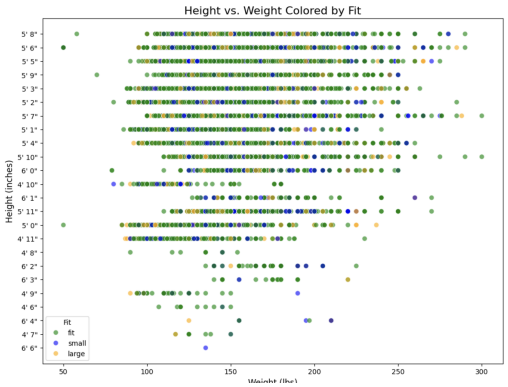

Figure 4: Plots - Correlation

for improvement, particularly for users at the extremes. The rating distribution in Figure 2 shows that the majority of ratings cluster around 10, reflecting positive user experiences, although ratings below 6 are rare and indicate occasional dissatisfaction. Figure 3, the correlation heatmap, reveals a strong positive correlation between weight and size (0.84) and a moderate correlation between weight and height (0.4), while rating has negligible correlation with these numerical features. Figure 4, a scatter plot of height versus weight colored by fit feedback, confirms that most users report a "fit" fit, but "small" and "large" fit issues tend to occur at the extremes of height and weight. Figure 5 further shows that height and weight are positively correlated, with "fit" feedback being most common, while "small" and "large" issues are more frequent at the outer ranges of weight and size. Age, however, does not show a strong relationship with fit feedback, indicating minimal impact on sizing.

In summary, the sizing system is largely effective, but improvements could be made for users at the extreme ends of size and weight. Height and weight have a clear relationship with fit feedback, with larger sizes and weights linked to more "small" or "large" fit issues. However, user satisfaction, as measured by ratings, does not appear to be significantly influenced by age or other numerical features, suggesting that other factors might play a role in determining overall satisfaction.

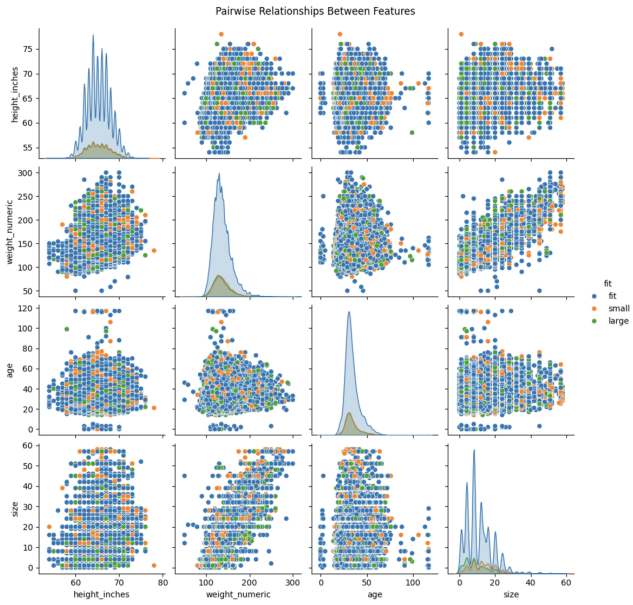

Figure 5: Plots - Pairwise Feature Relationships

1.3 Data Processing

To prepare the dataset for analysis and modeling, several non-numerical features were transformed into numerical labels. The bust size feature, typically represented in formats like 34D, was split into two separate numerical columns: band size, which captured the numeric portion (e.g., 34), and cup size, which encoded the letter component (e.g., D as a numerical value, where A=1, B=2, and so on). This transformation normalized the representation of bust sizes while retaining both components for analysis. Invalid or missing values were appropriately handled by assigning None.

The weight column, expressed in formats like 137lbs, was converted into a numerical column weight numeric by extracting the numeric portion and converting it into a floating-point value. Similarly, the height column, represented in a mixed format such as 5' 8", was standardized to a single numerical feature, height inches, where all heights were expressed entirely in inches (e.g., 5' 8" was converted to 68 inches). This ensured consistency and simplified further analysis.

Categorical features such as body type and rented for were also numerically encoded. Unique values in the body type column were mapped to corresponding numerical labels, creating the body type encoded column. Similarly, the rented for column, which described the context of the rental (e.g., wedding, vacation), was transformed into rented for encoded using a similar mapping approach. These transformations, along with the handling of missing values, resulted in a clean and consistent dataset that was ready for exploratory data analysis and machine learning model development.

The n-grams analysis highlights common words and phrases in review texts categorized by fit prediction (*fit*, *small*, and *large*). For the *fit* category, terms like "dress," "size," "fit perfectly," and "true size" dominate, indicating positive experiences. In the *small* category, phrases such as "runs small" and "tight" reflect dissatisfaction with undersized items. Similarly, in the *large* category, phrases like "runs large" and "little big" point to issues with oversized clothing. Across all categories, mentions of "many compliments" and "loved dress" suggest that despite fit issues, customers appreciated the product design. This analysis provides insights into user sentiments, useful for refining size recommendations and enhancing customer satisfaction.

Figure 6: Plots - N-grams Analysis for Fit Prediction

2 RELATED WORK

Product size recommendation has been an area of growing interest in e-commerce, especially in the fashion industry, where accurate size recommendations can significantly improve customer satisfaction and reduce return rates. Several approaches have been proposed to address the challenges associated with size prediction, customer feedback, and handling imbalanced data.

Abdulla and Borar [1] introduced a size recommendation system for fashion e-commerce, focusing on

predicting the right size for customers by considering both their preferences and product attributes. This approach combines collaborative filtering techniques with product information to provide accurate size recommendations. Sembium et al. [7] proposed a model for recommending product sizes by using customer fit preferences and product characteristics, integrating these factors into a unified recommendation framework. Their work emphasized the importance of personalized recommendations in the fashion domain.

In a similar vein, Sembium et al. [8] explored Bayesian models for product size recommendations, focusing on fitting product sizes to customer profiles. Their research leverages a probabilistic approach to predict the best size for customers, considering uncertainties and variations in their preferences.

Metric learning, a powerful technique for learning distance functions that can improve recommendation accuracy, has also been applied to this domain. Bellet et al. [2] provided an in-depth survey on metric learning, explaining how it can be used for feature vectors and structured data. Their work has been influential in adopting metric learning for tasks like recommendation systems. Huang et al. [4] explored deep representation learning for imbalanced classification tasks, applying it to size recommendation systems where the imbalance in size categories can lead to biased predictions. McFee and Lanckriet [5] applied metric learning to ranking tasks, demonstrating its ability to rank items based on user preferences, a technique that has been extended to size recommendation problems.

Misra et al. [6] proposed a framework for decomposing fit semantics in size recommendation systems. They used metric learning to capture fine-grained aspects of product fit and integrate these into a model that could more accurately recommend product sizes to customers. Their work also introduced public datasets that serve as valuable benchmarks for future research in this area.

Lastly, Hsieh et al. [3] introduced collaborative metric learning, which leverages user and item interaction data to create personalized recommendation systems. While their focus was on domains like music and online dating, their methodology has been adapted for fashion e-commerce, highlighting the flexibility and potential of metric learning-based approaches for size recommendations.

3 PREDICTIVE TASK: FIT PREDICTION ON RENT THE RUNWAY DATASET

Task Description

The task is to classify whether a clothing item will fit a user correctly, with the target variable *fit* having three classes: *small*, *fit*, and *large*. Predictions will be based on user attributes (e.g., height, weight, age) and product features (e.g., size).

Evaluation Metrics

- **Accuracy:** Percentage of correctly predicted instances.
- **F_2-score:** Weighted toward recall, critical for predicting the correct *fit*.

Baselines

The following baseline models will be used:

- **Random Guessing:** Uniform and weighted random predictions.

Model Validity Assessment

- **Balanced Datasets:** Minority classes will be balanced using:
  - **Downsampling:** Reducing majority class samples.
  - **Oversampling:** Increasing minority samples using duplication or synthetic methods.

Features and Preprocessing

Features Used:

- **User Attributes:** Height, weight, age, and body type.
- **Product Features:** Size and other categorical attributes.
- **Interaction Data:** User-item interactions for collaborative filtering.

Preprocessing:

- Missing values will be imputed using mean/median (numerical) or mode (categorical).
- Categorical features will be one-hot or label-encoded.
- Continuous variables will be normalized.

Models Used

The following models, relevant to course content, will be implemented:

- **Bayesian Personalized Ranking (BPR):** Collaborative filtering based on user-item interactions.
- **Logistic Regression:** A linear classifier.
- **Ensemble Learning:** Ensemble learning is a machine learning technique that combines multiple models to improve performance, reduce overfitting, and increase accuracy by leveraging the strengths of different algorithms.

The fit prediction task is a challenging multiclass classification problem with class imbalance. Implementing both baseline and advanced models with appropriate evaluation metrics will help identify the most effective approach. Balancing techniques and careful feature engineering will further enhance model performance.

4 EXPERIMENT DESIGN

4.1 Algorithm Design

Method 1 - Random Guessing

Random guessing, the simplest baseline, generates predictions without using dataset features. It evaluates advanced models by ensuring their predictive power exceeds chance. Two approaches, uniform and weighted random guessing, offer varying levels of sophistication.

Algorithm 1 - Uniform Random Guessing

For K unique labels, the probability of predicting any label k is:

$$p_k = \frac{1}{K}, \quad \text{for } k = 1, 2, \ldots, K$$

Predictions $\hat{y}$ are sampled as:

$$\hat{y} \sim \text{Uniform}(p_1, p_2, \ldots, p_K)$$

Algorithm 2 - Weighted Random Guessing

This algorithm enhances random guessing by incorporating label frequencies from the training set, aligning prediction probabilities with label distribution. This reduces bias in skewed datasets, making predictions more representative. While still random, it introduces a data-driven element, providing a stronger baseline for evaluating advanced models.

The detailed algorithm expression is here: he probability p_k of predicting a label k is derived from the training dataset as:

$$p_k = \frac{\text{count}(y_{\text{train}} = k)}{\text{total count}(y_{\text{train}})}, \quad \text{for } k = 1, 2, \ldots, K$$

Predictions $\hat{y}$ are sampled as:

$$\hat{y} \sim \text{Categorical}(p_1, p_2, \ldots, p_K)$$

Strengths

1) Baseline Benchmark: Provides a clear threshold for evaluating model performance.
3) Unbiased Representation: Matches the class distribution of the test set, avoiding bias from imbalanced predictions.

Weaknesses

1) No Data Utilization: Ignores dataset features entirely.
2) Overly Simplistic: Ineffective for datasets with skewed class distributions.

Method 2 - Naïve Bayes

The Naïve Bayes model is a probabilistic classifier based on Bayes' theorem, assuming conditional independence among features given the class label. This simplifies computations and makes it suitable for tasks requiring probabilistic reasoning when the independence assumption holds.

Algorithm - Gaussian Naïve Bayes

For continuous features, the likelihood $P(x_i|C)$ is modeled as a Gaussian distribution:

$$P(x_i|C) = \frac{1}{\sqrt{2\pi\sigma_C^2}} \exp\left(-\frac{(x_i - \mu_C)^2}{2\sigma_C^2}\right)$$

where:

- μ_C is the mean of feature x_i for class C.
- σ_C^2 is the variance of feature x_i for class C.

Strengths

1) Efficiency and Scalability: Fast computation, even on large datasets.
2) Effective for Continuous Features: Works well with numeric data under the normality assumption.

Weaknesses

1) Conditional Independence Assumption: Unrealistic

in many cases, potentially reducing accuracy.
2) No Feature Interaction: Ignores relationships between features, limiting its ability to capture complex patterns.

Method 3 - Logistic Regression Models

Logistic Regression is a popular classification algorithm that predicts the probability of a class based on a linear combination of input features. For the fit feedback task, it provides the advantage of interpretability, highlighting the features most strongly linked to the target variable.

General Description of Logistic Regression

Logistic regression predicts the probability of a class C given the input features $x_1, x_2, \ldots, x_n$ using the logistic function:

$$P(C|x) = \frac{1}{1 + \exp(-z)}, \quad \text{where } z = \beta_0 + \sum_{i=1}^{n} \beta_i x_i$$

Here:

- z: Linear combination of input features x_i with coefficients β_i.
- β_0: Intercept term.

Algorithm 1 - Logistic Regression Using All Features

This method uses all eight features: *weight*, *height*, *size*, *age*, *bandsize*, *cupsize*, *bodytype*, and $rented - forcategories$.

Algorithm 2 - Logistic Regression Using Top 5 Features

This method employs Recursive Feature Elimination (RFE) to identify the five most important features, enhancing model simplicity and focusing on key predictors. By limiting to the top five features, it reduces overfitting risk, improves interpretability, and balances simplicity with predictive performance.

Feature Selection and Performance Evaluation For feature selection, Recursive Feature Elimination (RFE) is used to rank features based on their importance.

Algorithm 3 - Logistic Regression Using Body-Related Features

This method uses body-related features only: *weight*, *height*, *size*, *bandsize*, *cupsize*, and *bodytype*. This algorithm selects body-related features, excluding non-physical attributes like age and rented-for category.

Strengths

1) Domain-Specific Focus: Prioritizes features directly related to physical fit for actionable insights.
2) Feature Selection: Enhances generalizability by excluding less relevant features, reducing dimensionality and simplifying interpretation.
3) Interpretable: Offers insights into the relationship between features and predictions through coefficients.

Weaknesses

1) Risk of Overfitting: Using all features can lead to overfitting if irrelevant ones are included.
2) Feature Dependency: Assumes independence among selected features, which may not hold.

Method 4 - Distance-Based Models k-NN

The k-Nearest Neighbors (k-NN) algorithm classifies instances based on the majority class among the k closest training samples, using Euclidean distance to measure similarity. This intuitive approach effectively classifies "fit feedback" by comparing feature similarity.

Classification Rule

For a given instance x, the predicted class $\hat{C}$ is:

$$\hat{C} = \arg\max_{C} \sum_{i \in N_k(x)} \mathbb{I}(y_i = C)$$

Algorithm 1 - k-NN Using All Features This method extends the feature set to include all eight features: *weight*, *height*, *size*, *age*, *bandsize*, *cupsize*, *bodytype*, and $rented - forcategory$.

Algorithm 2 - kNN Using Body-Related Features This approach uses the body-related features and size: *weight*, *height*, *bandsize*, *cupsize*, *bodytype*, and *size*. This algorithm builds on previous one by just including a less range of features.

Strengths

1) Versatile: Adapts to both linear and non-linear decision boundaries.
2) Interpretable: Predictions rely on direct similarity to training samples, aligning well with the fit feedback task.

Weaknesses

1) Computational Cost: Storing all training data and calculating distances for predictions can be resource-intensive, especially in higher dimensions.

Method 5 - Stochastic Gradient Descent (SGD)-Based Support Vector Machine (SVM)

The SGD-based Support Vector Machine (SVM) is a linear classifier that uses stochastic gradient descent to optimize the hinge loss function. This approach balances computational efficiency with the robust objective of SVM, making it ideal for large datasets and high-dimensional spaces.

For the fit feedback task, the model prioritizes body-related features—*weight*, *height*, *bandsize*, *cupsize*, *bodytype*, and *size*—as they are most relevant for predicting outcomes.

Hinge Loss Function The hinge loss function $L(\mathbf{w}, b)$ is defined as:

$$L(\mathbf{w}, b) = \frac{1}{N} \sum_{i=1}^{N} \max(0, 1 - y_i(\mathbf{w} \cdot \mathbf{x}_i + b)) + \frac{\lambda}{2} \|\mathbf{w}\|^2$$

where:

- $\mathbf{w}$: Weight vector.
- b: Bias term.
- y_i: True label (± 1).
- $\mathbf{x}_i$: Feature vector of the i-th instance.
- λ: Regularization parameter.
- $\|\mathbf{w}\|^2$: Regularization term, encouraging simpler models.

Strengths

1) Regularization: Reduces overfitting risk through built-in regularization techniques.

Weaknesses

1) Hyperparameter Sensitivity: Performance depends on careful tuning of parameters like learning rate and regularization strength.
2) Linear Assumption: Limited to linear decision boundaries, which may not suit non-linear data.

Method 6 - Latent Factor Model Using Matrix Factorization

This method combines matrix factorization with logistic regression to classify fit feedback. Truncated Singular Value Decomposition (SVD) is used to reduce high-dimensional body-related attributes into a lower-dimensional latent space.

Matrix Factorization Using Truncated SVD Given an input matrix $\mathbf{X} \in \mathbb{R}^{m \times n}$ with m samples and n features, the Truncated SVD factorizes $\mathbf{X}$ as:

$$\mathbf{X} \approx \mathbf{U}\Sigma\mathbf{V}^\top$$

where:

- $\mathbf{U} \in \mathbb{R}^{m \times k}$: Left singular vectors for the k latent factors.
- $\Sigma \in \mathbb{R}^{k \times k}$: Diagonal matrix of the top k singular values.
- $\mathbf{V} \in \mathbb{R}^{n \times k}$: Right singular vectors representing feature space.

The reduced representation $\mathbf{U}\Sigma$ captures the most significant variance in the data.

For Classification, we just use the Logistic Regression

Strengths

1) Dimensionality Reduction: Compresses high-dimensional data into a concise latent space, reducing noise and redundancy.
2) Generalizability: Lower-dimensional representations often enhance performance on unseen data.

Weaknesses

1) Loss of Interpretability: Latent features are not directly interpretable like original features.
2) Parameter Sensitivity: Performance depends on selecting the appropriate number of latent factors (n).

Method 7 - Gradient Boosting Models XGBoost

This approach employs XGBoost, a gradient boosting framework, to predict fit feedback using the provided features. Gradient Boosting models, like XGBoost, combine weak learners (typically decision trees) sequentially to minimize errors and enhance predictions. These models excel in structured data classification, effectively capturing complex feature interactions and handling both numerical and categorical data.

Gradient Boosting Framework The model at iteration m is given by:

$$F_m(x) = F_{m-1}(x) + \alpha h_m(x)$$

where:

- $F_m(x)$: Updated model at iteration m.
- $F_{m-1}(x)$: Model from the previous iteration.
- $h_m(x)$: Weak learner (e.g., decision tree).
- α: Learning rate controlling the contribution of $h_m(x)$ to the final model.

Strengths
1) Flexibility: Can handle a wide variety of data types and distributions.
Weaknesses

1) Overfitting Risk: Needs regularization and monitoring to mitigate overfitting.

Method 8 - Gradient Boosting Models CatBoost

CatBoost is a gradient boosting algorithm optimized for handling categorical features without extensive preprocessing. By ordering categories based on target statistics, it avoids one-hot encoding and manual transformations. In this approach, CatBoost leverages body-related and categorical features to predict fit feedback, making it ideal for structured data classification tasks.

Algorithm - CatBoost Using Selected Features

The classification rule for CatBoost involves aggregating the predictions from multiple decision trees. The final prediction is calculated as:

$$\hat{C} = \arg\max_{C} \left(\sum_{i=1}^{T} \alpha_i h_i(x) \right)$$

where:

- $h_i(x)$ is the output of the i-th tree,
- α_i is the weight associated with the i-th tree,
- T is the total number of trees.

Strengths
1) Efficient handling of categorical features.
2) High predictive accuracy due to gradient boosting.
3) Robust to overfitting with built-in regularization.

Weaknesses
1) Computationally intensive, especially with large datasets.
2) Requires careful tuning of hyperparameters.
3) Less interpretable due to the ensemble structure.

Method 9 - Random Forest

The Random Forest algorithm is an ensemble method that combines predictions from multiple decision trees to improve accuracy and robustness. By training T trees on bootstrap samples and using a random subset of f features for splits, it reduces overfitting and captures non-linear relationships. Predictions are made through a majority vote across all trees.

Strengths
1) Stability: Reduces variance, improving generalizability compared to single decision trees.
Weaknesses

1) Complexity: Higher computational cost than simpler models like logistic regression.
2) Interpretability: Aggregated predictions reduce transparency compared to single trees.

Method 10 - Single Decision Tree

This approach uses a single decision tree trained on body-related features to predict fit feedback. The Decision Tree algorithm partitions the feature space into regions using decision rules derived from training data, offering a simple and interpretable classification model.

Tree Construction The tree splits the data at each node to minimize Gini impurity:

$$\text{Gini}(t) = 1 - \sum_{i=1}^{C} p_i^2$$

where:

- p_i: Proportion of samples belonging to class i at node t.
- C: Number of classes.

Splits continue recursively until stopping criteria, such as maximum depth or minimum samples per leaf, are met.

Strengths
1) Non-Linear Relationships: Effectively models non-linear interactions between features.
Weaknesses

1) Overfitting: Single decision trees can overfit to the training data, especially when grown to full depth.

2) Limited Generalizability: Less capable of capturing complex patterns compared to ensemble methods.

Method 11 - MLP (Multi-layer Perceptron)

The Multi-Layer Perceptron (MLP) is a feedforward neural network with multiple layers of neurons. By utilizing hidden layers, it transforms input features into higher-level representations, enabling the learning of complex non-linear relationships.

Model Description For a training dataset $\mathbf{X}$ with m samples and n features, the MLP model is structured as follows:

- The model starts with an input layer, which accepts features like weight, height, band size, cup size, body type, and size.
- For the j-th neuron in the l-th hidden layer, the output is:

$$a_j^{(l)} = \sigma\left(\sum_{i=1}^{n} w_{ij}^{(l)} x_i + b_j^{(l)}\right)$$

  where $a_j^{(l)}$ is the output of the j-th neuron in the l-th layer, $w_{ij}^{(l)}$ are the weights, x_i are the inputs, $b_j^{(l)}$ is the bias term, and σ is the activation function.
- The final output layer used a softmax activation function:

$$\hat{y} = \text{softmax}\left(\sum_{i=1}^{n} w_i x_i + b\right)$$

  where w_i and b are the weights and bias for the output layer.

The model is trained by minimizing a loss function, typically cross-entropy for classification tasks, using optimization algorithms like stochastic gradient descent (SGD). During training, weights and biases are iteratively adjusted to reduce prediction error and improve model performance.

Strengths
1) Non-Linearity: Activation functions like ReLU enable modeling intricate input-output relationships.

Weaknesses
1) Overfitting: Prone to overfitting without adequate regularization, particularly with limited data.

Method 12 - Ensemble Learning (Stacking)

Stacking is an ensemble method that improves prediction accuracy by combining multiple base learners. Each base model is trained on the same dataset, and their predictions serve as input features for a meta-model, which makes the final decision.

Model Description For a training dataset $\mathbf{X}$ with m samples and n features, stacking involves the following steps:

- The base models used are:
  - XGBoost Classifier (xgb),
  - Random Forest Classifier (rf),
  - Multi-layer Perceptron Classifier (mlp).
- These predictions are then used as input features for the meta-model (a Logistic Regression classifier in this case).

Strengths
1) Flexibility: Works with diverse machine learning models, adaptable to various data types and tasks.

Weaknesses
1) Hyperparameter Sensitivity: Requires careful tuning of both base and meta-models for optimal results.

Method 13 - TabNet Model

TabNet is a deep learning model tailored for structured data, combining attention mechanisms with decision trees to learn high-level feature representations while preserving interpretability.

Model Description Given a training dataset $\mathbf{X}$ with m samples and n features:

- TabNet uses a deep learning architecture with a series of attention-based layers that select which features to focus on at each step.
- It employs a sequential attention mechanism to determine the most relevant features for each decision-making step in the model.
- The final prediction is made by aggregating the outputs from the attention layers, with a softmax function

Strengths
1) Feature Selection: The attention mechanism identifies key features, aiding in datasets with many variables.

Weaknesses
1) Computational Cost: Training requires significant resources, especially for large datasets.
2) Structured Data Focus: Performs best on structured data, with limited effectiveness on unstructured data without modifications.

Method 14 - Text-CNN Model

The Text-CNN model leverages Convolutional Neural Networks (CNNs) to classify fit feedback by extracting patterns from textual reviews. It enhances the traditional approach by incorporating review data, providing a richer representation of the problem.

Model Overview The Text-CNN architecture combines text mining and neural networks:

1) Embedding Layer: Transforms words into dense vectors encoding semantic and syntactic properties. Convolutional Layers: Use filters (e.g., 3, 4, 5) to extract n-gram features and local dependencies.
2) Pooling Layer: Aggregates key features, reducing dimensionality while retaining essential information.
3) Fully Connected Layer: Integrates features for the final classification.
4) Dropout: Mitigates overfitting by randomly deactivating neurons during training. This design captures textual patterns, such as phrases and sentiments, essential for interpreting review feedback.

Insights into Text Mining The Text-CNN model effectively extracts insights from textual reviews:
1) Phrase-Level Understanding: Captures patterns like "too tight" or "fits perfectly," directly influencing predictions.
2) Semantic Relationships: Embedding layers generalize by mapping semantically similar words (e.g., "tight" and "snug") to similar representations.
3) Feedback Integration: Aligns customer reviews with fit predictions, making the process more customer-focused.

Strengths
1) Textual Insights: Incorporates customer sentiment and detailed feedback into predictions.

Weaknesses

1) Data Quality Dependence: Performance is affected by noisy or irrelevant reviews.
2) Tokenization Constraints: Fixed input lengths may truncate longer reviews, losing context.

Method 15 - Bayesian Personalized Ranking

BPR is a ranking algorithm for recommendation systems that learns to rank items based on user preferences. It works well with implicit feedback, such as clicks, purchases, or views, rather than explicit ratings.

Model Description

For a dataset with m users and n items, BPR ranks items by:

- **Data Preparation:** Using triplets (u, i, j), where:
  - u: User ID.
  - i: Positive item (interacted with by the user).
  - j: Negative item (not interacted with).
- **Loss Function:** Optimizing a pairwise ranking loss:

$$\mathcal{L} = -\sum_{(u,i,j)} \log \sigma(\hat{x}_{ui} - \hat{x}_{uj}) + \lambda ||\Theta||^2$$

  where $\hat{x}_{ui}$ and $\hat{x}_{uj}$ are the predicted scores for items i and j, σ is the sigmoid function, and Θ represents model parameters.
- **Embedding Representation:** Latent embeddings represent users and items, learned during training.

Strengths

(1) **Adaptable:** Effective for implicit feedback and sparse datasets.
(2) **Scalable:** Handles large datasets efficiently.
(3) **Personalized:** Captures user-specific preferences.

Weaknesses

(1) **Feature-Limited:** Uses only user-item interactions, ignoring other features.
(2) **Binary Feedback:** Assumes interactions are binary, oversimplifying preferences.
(3) **Cold Start:** Struggles with new users or items without interactions.

BPR is effective for personalized ranking but requires adaptations or complementary models for feature-rich or multiclass datasets.

We analyzed the prediction distributions of the BPR model for three scenarios: unbalanced data, balanced data (downsampling), and balanced data (oversampling). These visualizations aimed to identify classification

thresholds, but the results were suboptimal. The BPR model relies solely on user-item interactions and does not incorporate additional features (e.g., user attributes or item properties), limiting its ability to capture the dataset's complexity.

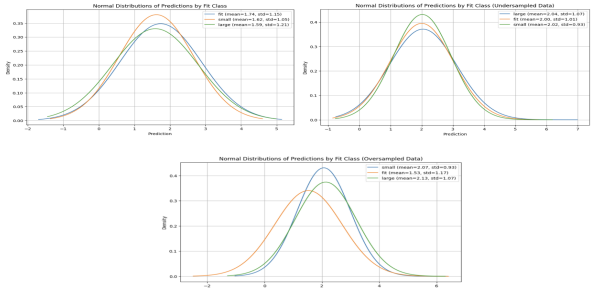

Figure 7: Plots - BPR distributions

4.2 Evaluation Metrics

For the evaluation of all models, we utilize the F_β score as the primary performance metric. The F_β score is a variant of the $F - score$ that places greater emphasis on recall compared to precision, making it particularly suitable for problems where correctly identifying all relevant instances is more critical than avoiding false positives.

In the context of the fit feedback classification task, recall is crucial because it ensures that feedback categories such as "Small" or "Large" are accurately captured, minimizing the risk of overlooking critical misfit cases that could lead to poor customer satisfaction. By prioritizing recall, the F_β score aligns with the objective of effectively addressing customer needs based on their feedback.

The weighted F_β score considers class imbalances, ensuring that performance is evaluated equitably across all feedback categories. This makes the F_β score a robust and reliable metric for assessing model performance in this context.

This metric provides an important overall view of how well the model is performing, particularly in terms of general classification.

The F_β score, with $\beta = 2$, is calculated as:

$$F_\beta = \left(1 + \beta^2\right) \cdot \frac{\text{precision} \cdot \text{recall}}{\beta^2 \cdot \text{precision} + \text{recall}}$$

where:

- precision is the ratio of correctly predicted positive observations to the total predicted positive observations.
- recall is the ratio of correctly predicted positive observations to all true positive observations.
- Both precision and recall are weighted by the class distribution in the test dataset.

In our evaluation, we also used the accuracy_score function from sklearn.metrics to compute the overall accuracy of the model. The accuracy_score metric calculates the proportion of correct predictions out of the total number of predictions made. Specifically, it is defined as the ratio of the number of correct predictions (where the predicted class matches the true class) to the total number of instances in the dataset. Mathematically, this is expressed as:

$$\text{Accuracy} = \frac{\text{Number of Correct Predictions}}{\text{Total Number of Predictions}}$$

By using both accuracy_score and the F_β score, we gain a more balanced perspective on the model's performance—accuracy provides an overall success rate, while the F_β score places more weight on correctly identifying the minority classes, which is particularly important in classification tasks with imbalanced data.

5 EVALUATION

Table 1 presents the performance comparison of various models across four evaluation metrics: unbalanced accuracy, unbalanced F_2-score, and balanced F_2-scores for downsampled and oversampled datasets. An ablation study was conducted, where minority classes were balanced using downsampling and oversampling, and models were re-evaluated.

Key Observations

(1) **Unbalanced vs. Balanced Performance:** Balanced F_2-scores were slightly lower than unbalanced F_2-scores for most models. However, some models, such as *Gradient Boosting (XGBoost)* and *CatBoost*, achieved their best results with oversampling.

(2) **Best Performing Model:** The *Text Mining-CNN Model* outperformed all other models across most metrics. It achieved the highest accuracy (0.8232) and unbalanced F_2-score (0.8151), as well as the best balanced F_2-scores of 0.7030 (downsampling)

and 0.8467 (oversampling). This indicates that the model is highly robust across both balanced and unbalanced datasets, benefiting significantly from oversampling.

(3) **Baseline Comparison:** The *Uniform Random Guessing Model* exhibited the poorest performance across all metrics. It achieved the lowest accuracy (0.3346), unbalanced F_2-score (0.3447), and balanced F_2-scores of 0.3254 (downsampling) and 0.3293 (oversampling). This highlights its ineffectiveness as a classifier and serves as a baseline for comparison.

(4) **Effect of Balancing:** While advanced models like *Text Mining-CNN* and *Random Forest* adapted well to balanced data, simpler models like *Naïve Bayes* and *k-NN* showed notable performance drops with balancing. Among ensemble methods, models like *Logistic Regression* and *Gradient Boosting* delivered balanced F_2-scores exceeding 0.44 in most cases.

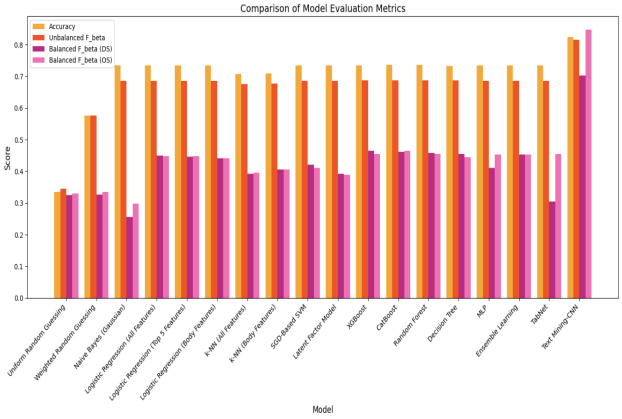

Figure 8: Plots - Comparison of Accuracy Metrics

6 CONCLUSION

The fit prediction task on the Rent the Runway dataset provides a challenging multiclass classification problem with class imbalance. By implementing both baseline and advanced models and using appropriate evaluation metrics, we aim to identify the most effective approach for this task. Additionally, balancing techniques and feature engineering ensure that the models utilize the dataset's full potential.

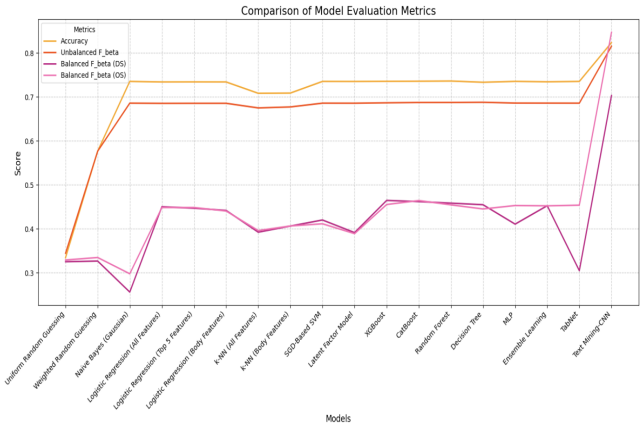

Figure 9: Plots - Comparison of Accuracy Metrics

Balancing the dataset influenced the performance of different models in varied ways. While some models experienced a decline in performance after balancing, others, such as *Text Mining-CNN* and tree-based methods, demonstrated improved results with balanced data. Importantly, the *Text Mining-CNN Model* outperformed all other models comprehensively, achieving the highest scores across all metrics. Almost all models outperformed the baseline methods, underscoring their ability to generalize better on this task. This highlights the importance of model selection and data preparation techniques, such as balancing, in achieving optimal results.

More sophisticated data balancing methods, such as SMOTE (Synthetic Minority Over-sampling Technique), could be explored to enhance model performance, particularly for underrepresented classes. Additionally, utilizing advanced evaluation metrics like precision-recall curves or class-specific metrics would provide a more nuanced understanding of the model's performance, especially for the minority classes. These metrics can help assess how well the model identifies the less frequent categories, offering deeper insights into areas that may require further improvement.

REFERENCES

[1] G Mohammed Abdulla and Sumit Borar. 2017. Size Recommendation System for Fashion E-commerce. *KDD Workshop on Machine Learning Meets Fashion* (2017).

[2] Aurélien Bellet, Amaury Habrard, and Marc Sebban. 2013. A survey on metric learning for feature vectors and structured data. *arXiv preprint arXiv:1306.6709* (2013).

Table 1: Model Performance Comparison across Different Sampling Methods

Model	Accuracy	Unbalanced F_β	Balanced F_β (DS)	Balanced F_β (OS)
Random Guessing:				
Uniform Random Guessing	0.3346	0.3447	0.3254	0.3293
Weighted Random Guessing	0.5767	0.5765	0.3269	0.3350
Naïve Bayes (Gaussian):	0.7351	0.6857	0.2564	0.2980
Logistic Regression Models:				
Using All Features	0.7339	0.6853	0.4503	0.4485
Using Top 5 Features (RFE)	0.7340	0.6854	0.4470	0.4486
Using Body-Related Features	0.7339	0.6854	0.4420	0.4406
Distance-Based Models k-NN:				
Using All Features	0.7081	0.6749	0.3925	0.3962
Using Body-Related Features	0.7086	0.6771	0.4064	0.4066
SGD-Based SVM:	0.7351	0.6857	0.4202	0.4116
Matrix Factorization Latent Factor Model:	0.7350	0.6856	0.3916	0.3892
Gradient Boosting Models XGBoost:	0.7353	0.6865	0.4647	0.4554
CatBoost Classifier	0.7356	0.6872	0.4620	0.4644
Random Forest:	0.7360	0.6873	0.4585	0.4546
Single Decision Tree:	0.7332	0.6877	0.4549	0.4453
Multilayer Perceptron	0.7352	0.6859	0.4108	0.4531
Ensemble learning	0.7343	0.6858	0.4527	0.4524
TabNet	0.7351	0.6857	0.3050	0.4539
Bayesian Personalized Ranking (BPR)	0.3532	0.7207	0.7000	0.6800
Text Mining-CNN Model:	0.8232	0.8151	0.7030	0.8467

[3] Cheng-Kang Hsieh, Longqi Yang, Yin Cui, Tsung-Yi Lin, Serge Belongie, and Deborah Estrin. 2017. Collaborative metric learning. In *Proceedings of the 26th International Conference on World Wide Web*.

[4] Chen Huang, Yining Li, Chen Change Loy, and Xiaoou Tang. 2016. Learning deep representation for imbalanced classification. In *CVPR*.

[5] Brian McFee and Gert R Lanckriet. 2010. Metric learning to rank. In *ICML*.

[6] Rishabh Misra, Mengting Wan, and Julian McAuley. 2018. Decomposing fit semantics for product size recommendation in metric spaces. In *Proceedings of the 12th ACM conference on recommender systems*. ACM, 422–426.

[7] Vivek Sembium, Rajeev Rastogi, Atul Saroop, and Srujana Merugu. 2017. Recommending Product Sizes to Customers. In *RecSys*.

[8] Vivek Sembium, Rajeev Rastogi, Lavanya Tekumalla, and Atul Saroop. 2018. Bayesian Models for Product Size Recommendations. In *Proceedings of the 2018 World Wide Web Conference on World Wide Web*.