**Makeup Recommendation Model Documentation**

**Introduction**

The Makeup Recommendation Model is designed to suggest the best makeup look based on a user’s features, including eye color, skin color, hair type, face shape, and more. This model uses **XGBoost** (Extreme Gradient Boosting), a powerful machine learning algorithm, to make personalized makeup suggestions with high accuracy.

This document outlines the libraries used, methods applied, expected outcomes, feature-label relationships, model configuration, and other key components for understanding the model.

**1. Libraries Used and Their Purpose**

| **Library** | **Purpose** |
| --- | --- |
| **pandas** | For handling dataset loading, manipulation, and preprocessing. |
| **imblearn.over\_sampling.SMOTE** | To balance the dataset by generating synthetic samples for underrepresented classes. |
| **sklearn.model\_selection.train\_test\_split** | For splitting the dataset into training and testing sets. |
| **sklearn.metrics** | To evaluate the model’s performance using accuracy score, classification reports, and confusion matrix. |
| **xgboost** | To build the makeup recommendation model using gradient boosting. |
| **matplotlib.pyplot & seaborn** | For visualizing data insights, such as confusion matrices and feature importance. |
| **joblib** | For saving and loading the trained model and label encoders for future use. |

**2. Why Use XGBoost?**

**XGBoost** is used for the following reasons:

* It is highly efficient, fast, and optimized for large datasets.
* It prevents overfitting through regularization techniques.
* It can model complex, non-linear relationships between features.
* It supports parallel processing, making the training process faster.
* It performs well in classification tasks with structured data, like makeup recommendations.

**3. Why Use SMOTE for Data Balancing?**

**SMOTE** (Synthetic Minority Over-sampling Technique) is applied to balance the dataset by:

* Generating synthetic samples for underrepresented makeup types.
* Ensuring the model does not bias its predictions toward more frequent classes.
* Improving model performance and generalization, especially for less common makeup looks.

**4. Methods Used in the Model**

**4.1 Data Preprocessing**

* **Missing Value Imputation**: Handled using SimpleImputer with the "most frequent" strategy for categorical features.
* **Label Encoding**: Converts categorical features (e.g., makeup type, skin tone) into numerical format using **LabelEncoder()**.
* **Feature Scaling**: Normalizes numerical data (if present) using a scaler to ensure model performance is not skewed by magnitude differences in data.

**4.2 Hyperparameter Tuning**

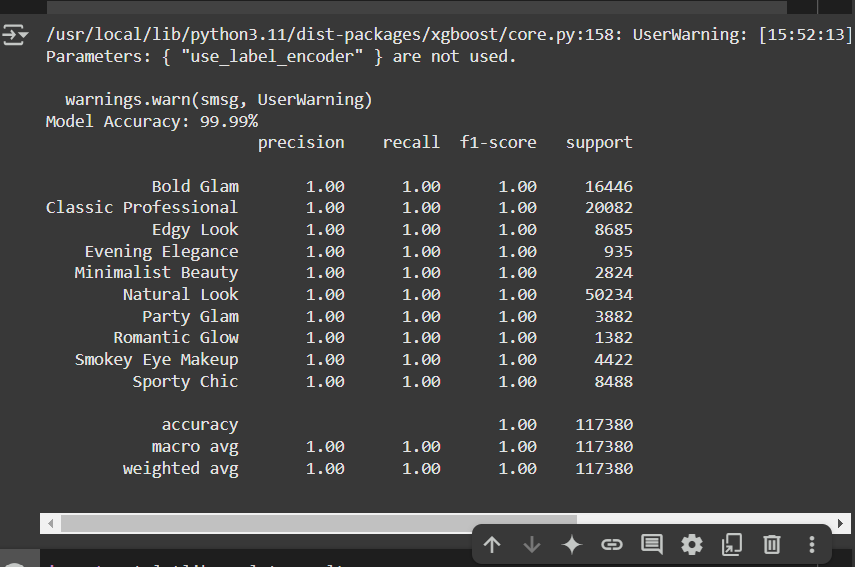
Manual fine-tuning of hyperparameters ensures optimal model performance:

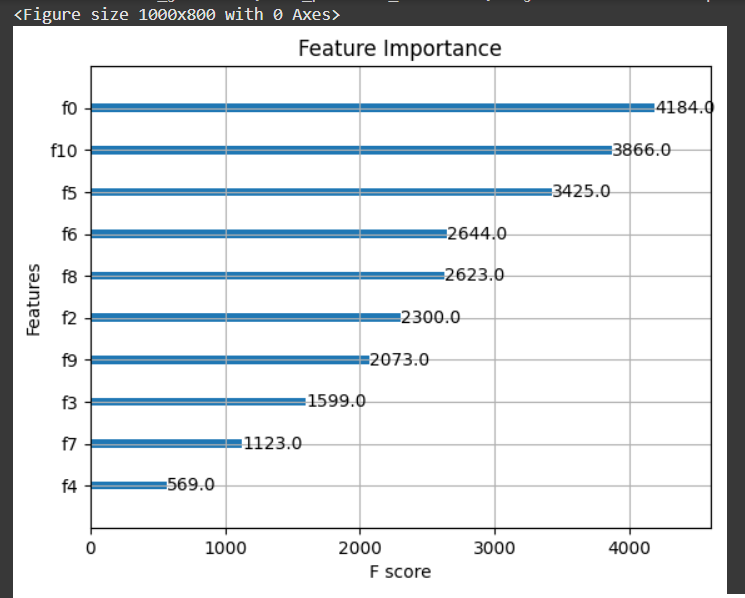
| **Hyperparameter** | **Purpose** |
| --- | --- |
| **n\_estimators** | Defines the number of trees in the model. |
| **max\_depth** | Controls the depth of each tree (higher values may cause overfitting). |
| **learning\_rate** | Determines the step size during optimization (lower values improve generalization). |
| **subsample** | Fraction of data used for each boosting round, helps prevent overfitting. |
| **colsample\_bytree** | Fraction of features used per tree to enhance diversity. |

**5. Expected Output of the Model**

The model will recommend the best makeup look based on the user's input features:

* **Makeup Look Name** (e.g., "Bold Glam", "Classic Professional")
* **Makeup Description** (e.g., "A vibrant and bold makeup style for an evening out.")
* **Accuracy**: The model is expected to achieve a high accuracy rate (above 90%) after proper tuning.







**6. How Features Work with Labels**

Each feature influences the makeup recommendations:

| **Feature** | **Impact on Makeup Recommendation** |
| --- | --- |
| **Eye Color** | Determines eye makeup shades that complement eye tones. |
| **Skin Color** | Influences foundation shades and overall makeup palette. |
| **Hair Type** | Affects styling choices (e.g., makeup look for curly vs straight hair). |
| **Face Shape** | Helps decide contouring and blush application techniques. |
| **Occupation** | Professional or bold makeup choices based on the user’s job. |
| **Eyebrow Shape** | Influences eyebrow makeup (e.g., bold vs natural fill). |
| **Lip Shape** | Affects lipstick shape and style recommendations. |
| **Look Type** | Determines if the user prefers a bold, natural, or classic look. |
| **Occasion** | Suggests makeup based on casual, formal, or special occasion needs. |

**7. Model Configuration**

* **Training Dataset**: 80%
* **Testing Dataset**: 20%
* **Oversampling**: SMOTE applied to balance classes
* **Algorithm**: XGBoost
* **Performance Metrics**:
  + Accuracy
  + Confusion Matrix
  + Classification Report

**8. Saving and Loading the Model**

The trained model and label encoders are saved using **joblib**, allowing the model to be deployed without retraining:

* **makeup\_recommendation\_model.pkl** → Stores the trained model.
* **label\_encoders.pkl** → Stores label encoders for categorical features.
* **scaler.pkl** → Stores the scaler for numerical features (if applicable).

These files make it easy to integrate the model into web or mobile applications for real-time predictions.

**9. Making Predictions with New Data**

* New user attributes are encoded using the saved **label\_encoders.pkl**.
* The model predicts the best makeup look based on these encoded features.
* The makeup name and description are retrieved from a predefined dictionary. Example Output:
* **Prediction 1**: Bold Glam - A vibrant and bold makeup look perfect for evening events.
* **Prediction 2**: Classic Professional - A subtle and refined look ideal for the workplace.
* **Prediction 3**: Edgy Look - A daring and modern look for creative individuals.

**10. Summary**

* The model recommends makeup looks based on various user features (eye color, skin tone, occupation, etc.).
* **XGBoost** provides fast and accurate recommendations.
* **SMOTE** helps balance the dataset for better generalization.
* Hyperparameter tuning ensures the model performs optimally.
* The model is saved and can be reused for future predictions in real-world applications.

**11. Future Improvements**

* Enhance model accuracy by incorporating additional features (e.g., age, facial expressions).
* Expand the dataset with more diverse makeup styles.
* Explore deep learning approaches for better feature extraction.
* Develop a mobile app for easy user access and personalized recommendations.

**12. Questions**

💡 **Q1: Why do we encode categorical data?**  
📝 Machine learning models require numerical data for processing, so categorical data is converted using label encoding.

💡 **Q2: Why do we use XGBoost instead of other models?**  
📝 XGBoost is known for its speed, accuracy, and ability to prevent overfitting, making it ideal for classification tasks like makeup recommendations.

💡 **Q3: Why is the data split into training and testing sets?**  
📝 To assess how well the model generalizes to new, unseen data and ensure it’s not overfitting.

💡 **Q4: Why do we save the model?**  
📝 To reuse the trained model in future predictions without needing to retrain it.

💡 **Q5: What happens if we don’t balance the dataset?**  
📝 The model may bias its predictions toward more frequent makeup styles, resulting in lower fairness and generalization.

💡 **Q6: Can this model work for all skin tones and makeup styles?**  
📝 Yes, as long as the dataset is diverse and inclusive of various skin tones, styles, and features.

💡 **Q7: Can this model be improved?**  
📝 Yes, by adding more features, tuning the hyperparameters further, and considering deep learning for better feature extraction.