Course: Machine Learning

Assignment: Final Assignment

Name: Sandip Magar

Date: 01/03/2025

Student ID: 23189646/2

Contents

Project Overview

Problem Definition:

The problem we are addressing is understanding the factors that contribute to hair fall. Hair fall can be influenced by various health metrics such as protein levels, keratin levels, vitamin levels, and stress levels. By analyzing these factors, we aim to identify potential correlations and patterns that could help in predicting or mitigating hair fall.

Real-World Significance:

Hair loss is a very common condition which bothers a large percentage of people. Analyzing the health parameters responsible for hair loss can assist in creating treatments and prevention strategies. It can be highly beneficial to healthcare providers, dermatologists, and people seeking to improve the health of their hair.

Dataset Description:

The Hair Loss Dataset is a well-curated collection designed to facilitate analysis and prediction of patterns of hair fall based on various biochemical, physical, and lifestyle factors. The dataset can prove particularly useful to researchers and data scientists interested in understanding the multifactorial causation of hair loss and creating predictive models to assess risk.

Dataset Structure and Columns

The dataset consists of the following columns:

1. total_protein

This column is the total protein level recorded in the individual's body. Proteins are important for the structure and development of hair, and their imbalance can result in hair loss.

Adequate protein is required for keratin synthesis, which is one of the predominant elements of hair. The total protein measures provide information on the overall nutritional level in relation to hair.

2.total_keratine

This column indicates the amount of keratin, the primary structural protein in hair.

The amount of keratin directly affects the strength and resilience of hair. Low keratin may cause brittle hair shafts and unnecessary hair loss.

3.hair_texture

Denotes qualitative measurement of hair texture, which can be graded as fine, medium, or coarse.

Hair texture can affect how hair responds to stress, treatments, and external conditions. An understanding of texture variations is required for personalized hair care recommendations.

4. vitamin

This column captures data regarding vitamin levels in the body, which are required for various metabolic processes like hair growth.

Vitamins such as biotin, vitamin D, and others play a role in healthy hair. Lack or imbalance can cause hair loss.

5.manganese

Shows the level of manganese in the body.

Manganese is a trace element involved in many enzymatic reactions, of which some play a role in hair health. Deficiency as well as toxicity can affect cycles of hair growth.

6.iron

It shows the level of iron, a critical mineral for oxygenation and cellular metabolism.

Iron deficiency is a frequent underlying cause of hair loss, particularly among women. Monitoring iron levels contributes to the understanding of potential anemia-related hair loss.

7.calcium

Referring to the concentration of calcium, another important mineral in the body.

Significance: Calcium participates in various cellular activities. Calcium balance is not only important for bones but also for hair follicle function.

8.body_water_content

This column refers to the overall water content within the body.

Adequate hydration is fundamental to all metabolic processes, such as those responsible for normal hair growth. Dehydration can result in impaired hair structure and hair loss.

9.stress_level

Assesses of self-reported or numerical stress levels reported by the patient.

Stress has been implicated to worsen hair condition by causing induction of hormonal disturbance and inflammatory response. This value is crucial to use in connecting psychological with physiological hair loss manifestations.

10.liver_data

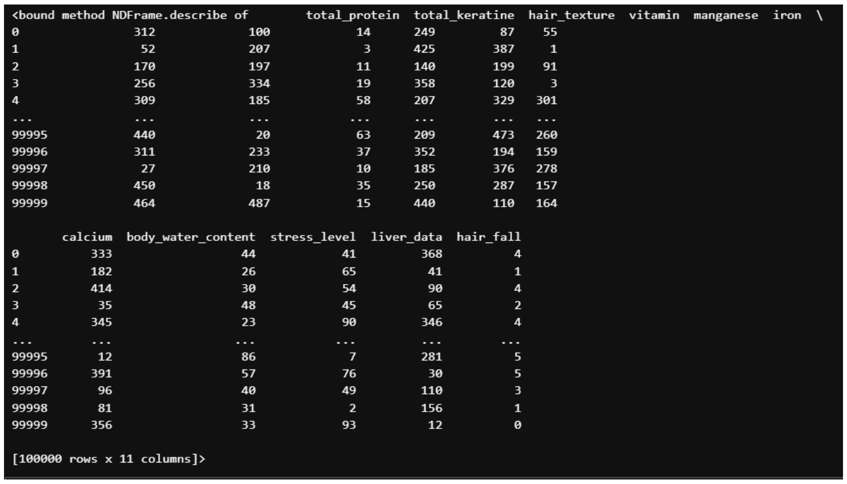
Contains information related to liver functioning or status.

Liver plays a pivotal role in metabolizing nutrients as well as toxins. Injured liver function could affect the body's overall state of health, including hair maintenance and growth processes.

11.hair_fall

The focus variable indicating the rate or the degree of loss of hair by the subject.

It is very important for the construction of predictive models. It is the primary outcome variable to establish the relationship between the measured lifestyle, physical, and biochemical variables and the degree of hair loss.



```
<bound method NDFrame.describe of        total_protein  total_keratine  hair_texture  vitamin  manganese  iron  \
0                312             100            14      249         87    55
1                 52             207             3      425        387     1
2                170             197            11      140        199    91
3                256             334            19      358        120     3
4                309             185            58      207        329   301
...              ...             ...           ...      ...        ...   ...
99995            440              20            63      209        473   260
99996            311             233            37      352        194   159
99997             27             210            10      185        376   278
99998            450              18            35      250        287   157
99999            464             487            15      440        110   164

       calcium  body_water_content  stress_level  liver_data  hair_fall
0          333                  44            41         368          4
1          182                  26            65          41          1
2          414                  30            54          90          4
3           35                  48            45          65          2
4          345                  23            90         346          4
...        ...                 ...           ...         ...        ...
99995       12                  86             7         281          5
99996      391                  57            76          30          5
99997       96                  40            49         110          3
99998       81                  31             2         156          1
99999      356                  33            93          12          0

[100000 rows x 11 columns]>
```

Fig1: Dataset Description

Justification for Dataset Choice:

This dataset is suitable for the problem as it includes a variety of health metrics that are known to influence hair health. The presence of both nutritional and stress-related metrics provides a comprehensive view of potential factors affecting hair fall.

Exploratory Data Analysis

Exploratory Data Analysis (EDA) is the process of analyzing and visualizing data to understand its structure, detect patterns, identify anomalies, and summarize key insights before building a machine learning model. It involves various statistical and graphical techniques to extract useful information from the dataset.

Histogram for numerical features

The purpose of plotting histograms for numerical features is to analyze the distribution of data, detect potential skewness, outliers, and data imbalances. These visualizations help in understanding the spread and concentration of values, allowing for better feature engineering and preprocessing decisions. In this case, the histograms provide insights into whether the features are uniformly distributed, have missing values, or exhibit patterns that could affect model performance.

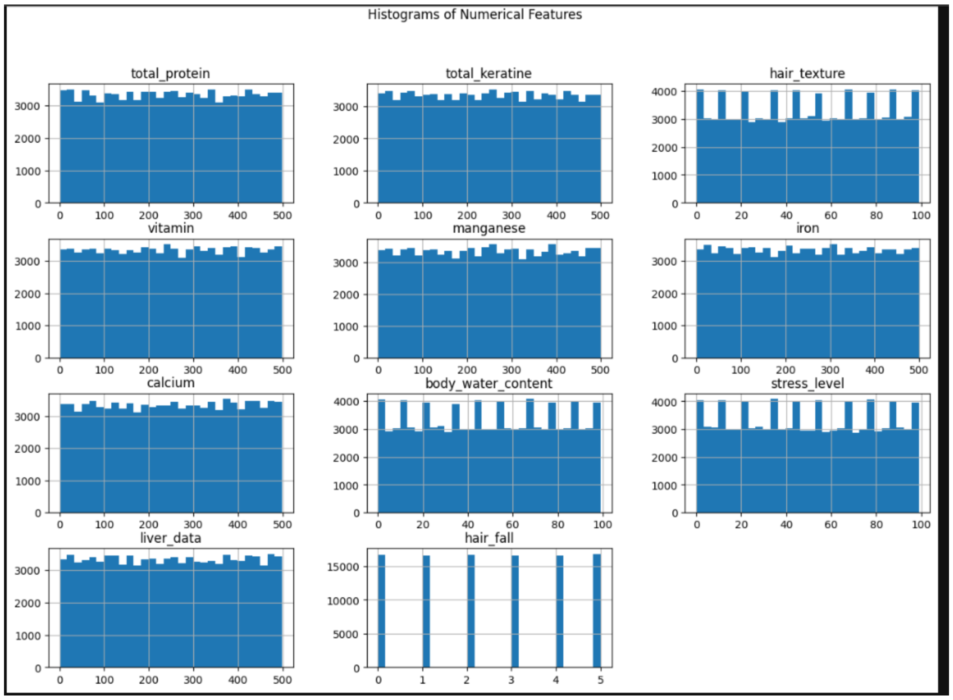


Fig2: Histogram for numerical features

Box plot to detect outliers

The box plots visualize the distribution of numerical features across different hair fall levels, helping to identify variations, outliers, and potential relationships between features and the target variable. Each box plot displays the median, interquartile range (IQR), and possible outliers for a given feature, categorized by hair fall levels.

From these plots, we can observe whether any feature exhibits noticeable differences in distribution across hair fall categories. If the medians and spreads of the boxes change significantly, it suggests a potential correlation between that feature and hair fall. However, if the distributions remain similar, it indicates weak or no association with hair fall.

This analysis is useful for feature selection, helping determine which variables might be informative for predictive modeling.

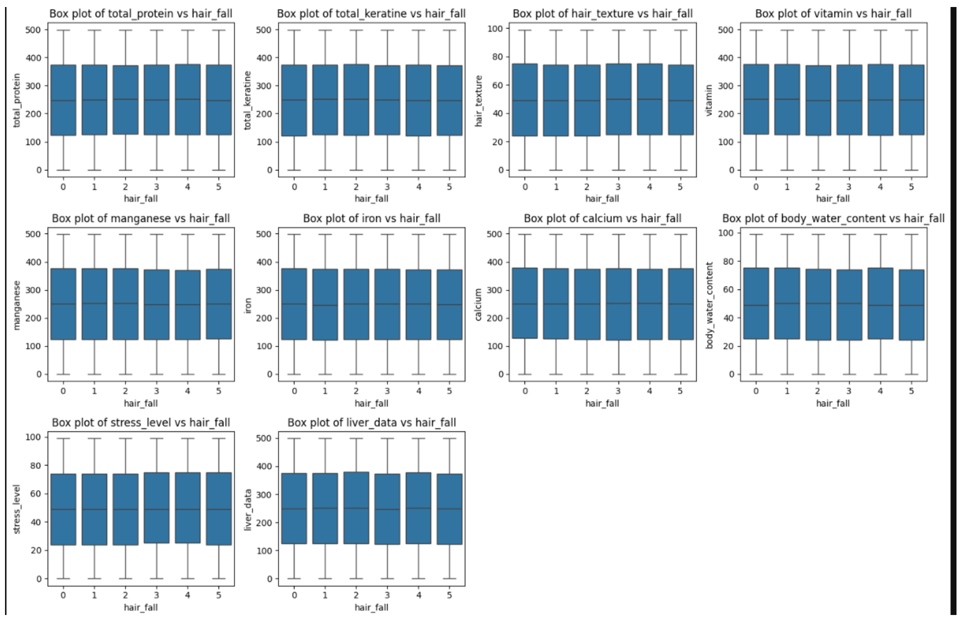


Fig3: Box plot

Correlation Matrix Heatmap

This heatmap represents the correlation matrix for the dataset, illustrating the relationships between numerical features, including the target variable, 'hair_fall.' The correlation values range from -1 to 1.

From the heatmap, all correlation values are very close to zero, suggesting that there are **weak or no significant linear relationships** between the features and 'hair_fall.' This indicates that individual features may not strongly influence hair fall when considered in isolation. These findings can inform feature selection and modeling decisions, suggesting the need for **non-linear models or feature engineering techniques** to uncover hidden patterns in the data.

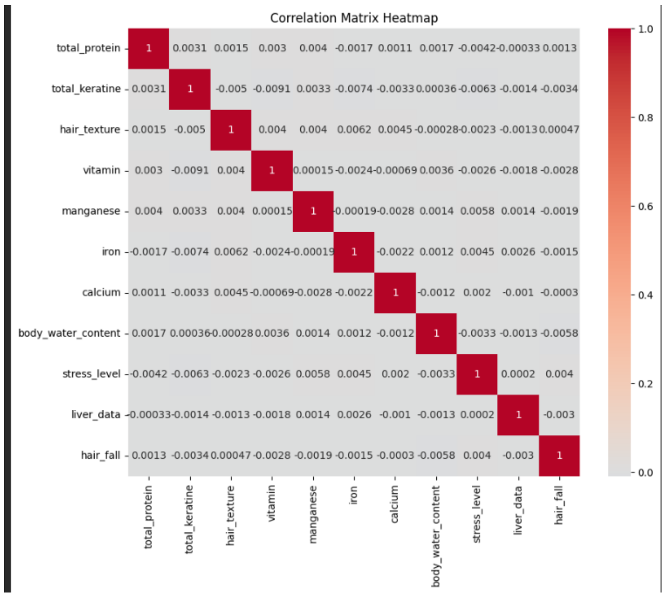


Fig4: Correlation matrix Heatmap

Correlation with hairfall

The given bar chart illustrates the correlation of various features with hair fall. Correlation measures the strength and direction of the relationship between independent variables (features) and the dependent variable (hair fall). Positive values indicate a direct relationship, meaning an increase in the feature may lead to higher hair fall, while negative values indicate an inverse relationship.

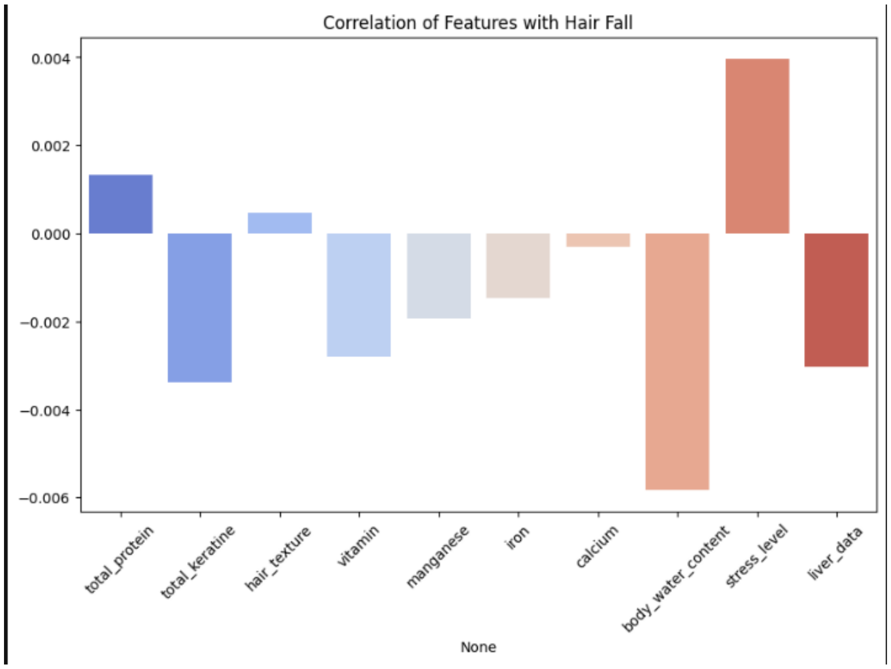
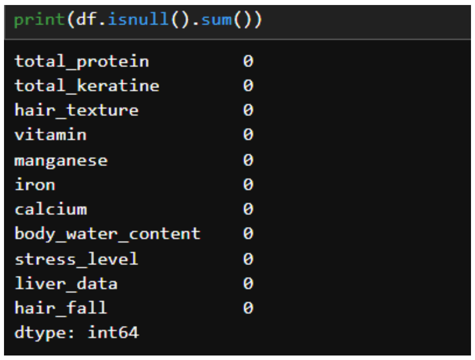


Fig5: Correlation with hairfall

Data Preprocessing and Feature Engineering

Preprocessing Steps

1. **Handling Missing Values**
   - No missing values were found in the dataset.

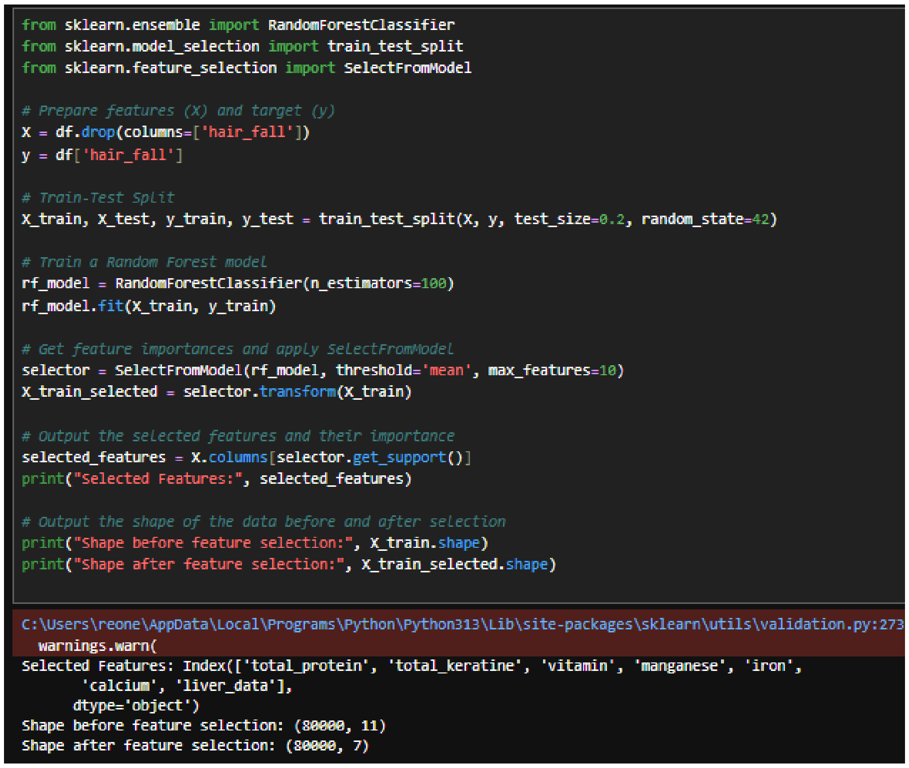


```
print(df.isnull().sum())

total_protein         0
total_keratine        0
hair_texture          0
vitamin               0
manganese             0
iron                  0
calcium               0
body_water_content    0
stress_level          0
liver_data            0
hair_fall             0
dtype: int64
```

Fig6: Missing Values

2. **Encoding Categorical Variables**
    - Observation: The dataset primarily consists of numerical features in this iteration.
    - Action: Checked for hair_texture column as a categorical and encoded. No encoding was performed as there are currently no categorical variables.
3. **Feature Scaling**
    - Rationale: Models like Logistic Regression and K-Nearest Neighbors are sensitive to feature magnitudes.
4. **Feature Selection**
    - Approach:
        1. Computed feature importance using a Random Forest classifier.
        2. Identified features with near-zero or very low importance for potential removal.
    - Findings: The following features exhibited similar values of importance and were further analyzed for redundancy:
        - total_keratine: 0.1421
        - manganese: 0.1427
        - iron: 0.1428
        - vitamin: 0.1429
        - liver_data: 0.1430
        - calcium: 0.1431
        - total_protein: 0.1435



```python
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
from sklearn.feature_selection import SelectFromModel

# Prepare features (X) and target (y)
X = df.drop(columns=['hair_fall'])
y = df['hair_fall']

# Train-Test Split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Train a Random Forest model
rf_model = RandomForestClassifier(n_estimators=100)
rf_model.fit(X_train, y_train)

# Get feature importances and apply SelectFromModel
selector = SelectFromModel(rf_model, threshold='mean', max_features=10)
X_train_selected = selector.transform(X_train)

# Output the selected features and their importance
selected_features = X.columns[selector.get_support()]
print("Selected Features:", selected_features)

# Output the shape of the data before and after selection
print("Shape before feature selection:", X_train.shape)
print("Shape after feature selection:", X_train_selected.shape)
```

```
C:\Users\reone\AppData\Local\Programs\Python\Python313\Lib\site-packages\sklearn\utils\validation.py:273
  warnings.warn(
Selected Features: Index(['total_protein', 'total_keratine', 'vitamin', 'manganese', 'iron',
       'calcium', 'liver_data'],
      dtype='object')
Shape before feature selection: (80000, 11)
Shape after feature selection: (80000, 7)
```

Fig7: Feature Selection

Justification of Feature Engineering and Selection

1. Model Generalization
   - The chosen transformations (e.g., scaling, feature selection) help ensure the model generalizes well to new data rather than overfitting to the training set.
2. Dimensionality Reduction and Interpretability
   - Removing or merging features with similar or negligible importance helps reduce model complexity.
   - A smaller set of relevant features improves model interpretability and potentially its predictive performance.
3. Consistent Feature Magnitude
   - Min-Max Scaling ensures features are on a similar scale, which is critical for distance-based algorithms like K-Nearest Neighbors and can help gradient-based optimizers converge more efficiently in models such as Logistic Regression.

Model Development and Evaluation

Model Selection and Implementation

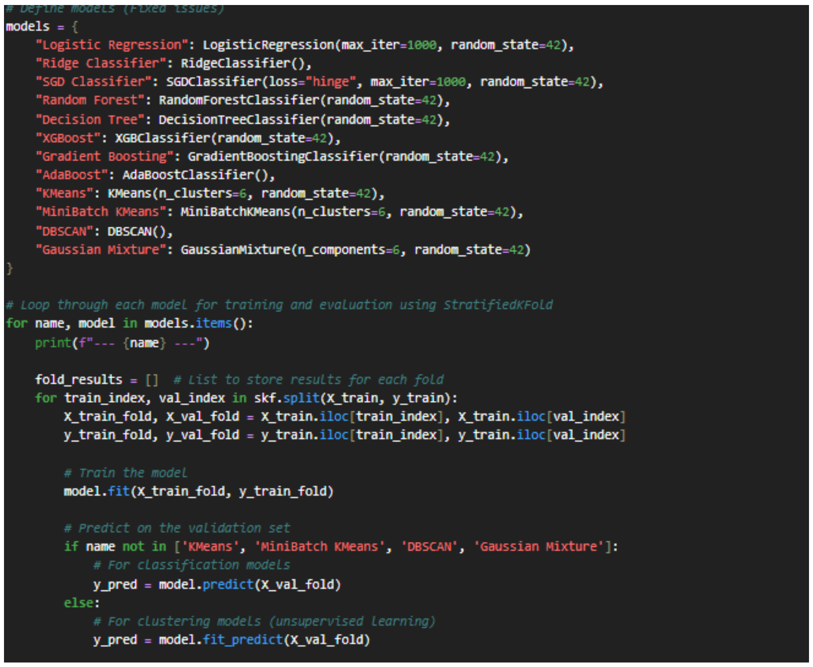
To ensure a comprehensive evaluation, a diverse set of machine learning models were implemented, covering both supervised and unsupervised learning approaches.

Supervised Learning Models

1. Logistic Regression - A linear model for binary/multiclass classification.
2. Ridge Classifier - A variation of logistic regression with L2 regularization.
3. SGD Classifier - A stochastic gradient descent-based classifier.
4. Decision Tree Classifier - A simple tree-based model for classification.
5. Random Forest Classifier - An ensemble learning method combining multiple decision trees.
6. Gradient Boosting Classifier - A boosting-based ensemble method.
7. AdaBoost Classifier - An adaptive boosting-based classifier.
8. XGBoost Classifier - A highly efficient gradient boosting classifier.

Unsupervised Learning Models

1. K-Means Clustering - A centroid-based clustering algorithm.
2. MiniBatch K-Means - A faster variant of K-Means.
3. DBSCAN - A density-based clustering algorithm.
4. Gaussian Mixture Model (GMM) - A probabilistic model that assumes data is generated from multiple Gaussian distributions.



```python
# Define models (Fixed issues)
models = {
    "Logistic Regression": LogisticRegression(max_iter=1000, random_state=42),
    "Ridge Classifier": RidgeClassifier(),
    "SGD Classifier": SGDClassifier(loss="hinge", max_iter=1000, random_state=42),
    "Random Forest": RandomForestClassifier(random_state=42),
    "Decision Tree": DecisionTreeClassifier(random_state=42),
    "XGBoost": XGBClassifier(random_state=42),
    "Gradient Boosting": GradientBoostingClassifier(random_state=42),
    "AdaBoost": AdaBoostClassifier(),
    "KMeans": KMeans(n_clusters=6, random_state=42),
    "MiniBatch KMeans": MiniBatchKMeans(n_clusters=6, random_state=42),
    "DBSCAN": DBSCAN(),
    "Gaussian Mixture": GaussianMixture(n_components=6, random_state=42)
}

# Loop through each model for training and evaluation using StratifiedKFold
for name, model in models.items():
    print(f"--- {name} ---")

    fold_results = []  # List to store results for each fold
    for train_index, val_index in skf.split(X_train, y_train):
        X_train_fold, X_val_fold = X_train.iloc[train_index], X_train.iloc[val_index]
        y_train_fold, y_val_fold = y_train.iloc[train_index], y_train.iloc[val_index]

        # Train the model
        model.fit(X_train_fold, y_train_fold)

        # Predict on the validation set
        if name not in ['KMeans', 'MiniBatch KMeans', 'DBSCAN', 'Gaussian Mixture']:
            # For classification models
            y_pred = model.predict(X_val_fold)
        else:
            # For clustering models (unsupervised learning)
            y_pred = model.fit_predict(X_val_fold)
```

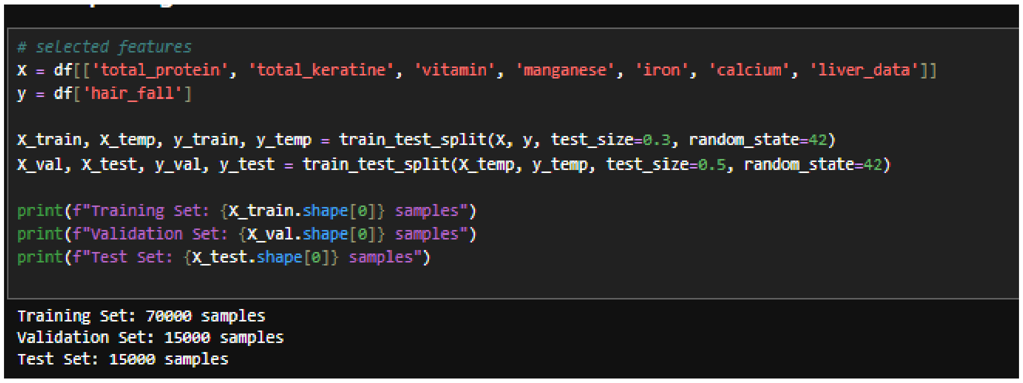
Fig8: Model Selection

Data Splitting Strategy

To ensure fair evaluation and avoid overfitting, the dataset was split as follows:

- Training Set (70%) - Used to train models.
- Validation Set (15%) - Used for hyperparameter tuning and model selection.
- Test Set (15%) - Used to evaluate final model performance.

Additionally, Stratified K-Fold Cross-Validation (5 folds) was implemented to ensure robustness by reducing variability in model performance.



```python
# selected features
X = df[['total_protein', 'total_keratine', 'vitamin', 'manganese', 'iron', 'calcium', 'liver_data']]
y = df['hair_fall']

X_train, X_temp, y_train, y_temp = train_test_split(X, y, test_size=0.3, random_state=42)
X_val, X_test, y_val, y_test = train_test_split(X_temp, y_temp, test_size=0.5, random_state=42)

print(f"Training Set: {X_train.shape[0]} samples")
print(f"Validation Set: {X_val.shape[0]} samples")
print(f"Test Set: {X_test.shape[0]} samples")
```

```
Training Set: 70000 samples
Validation Set: 15000 samples
Test Set: 15000 samples
```

Fig9: Data Split

Model Training and Performance Evaluation

Each supervised model was trained using Stratified K-Fold Cross-Validation (5 folds). The key performance metrics used for evaluation include:

- Precision: Measures how many of the predicted positive cases were actsually correct.
- Recall: Measures of how well the model identifies actual positive cases.
- F1-score: Harmonic mean of precision and recall, providing a balanced measure.
- Confusion Matrix: Helps in analyzing misclassifications.

Cross-Validation Results

For each model, computed the average classification report across all folds.

Confusion Matrix Analysis

The average confusion matrix was computed across all folds to analyze misclassification patterns.

Comparison of Supervised and Unsupervised Models

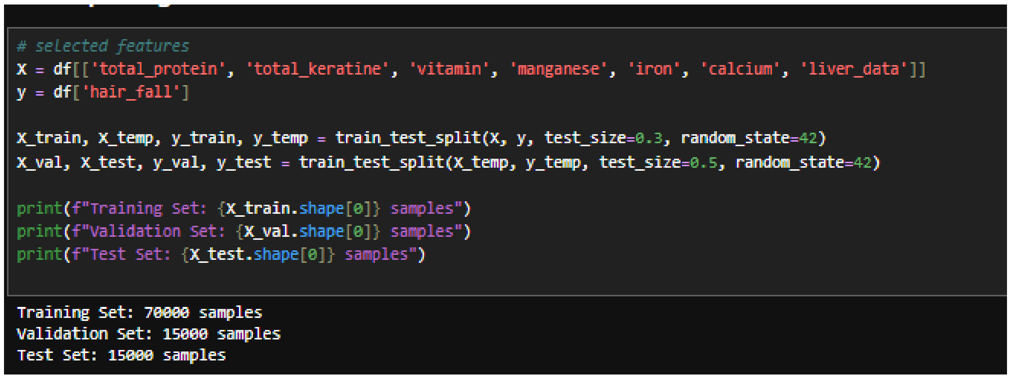
- Supervised models (e.g., XGBoost, Random Forest, Gradient Boosting) outperformed unsupervised models in terms of classification metrics.
- Unsupervised models (K-Means, DBSCAN, GMM) provided insights into natural groupings within the data but were less effective for direct classification.
- Tree-based models (Random Forest, Gradient Boosting, XGBoost) consistently performed better due to their ability to capture complex relationships.

Model Trade-offs and Selection

- Logistic Regression & Ridge Classifier performed well on linearly separable data but struggled with complex patterns.
- Decision Tree was prone to overfitting.
- Random Forest & XGBoost provided the best balance between precision, recall, and F1-score.
- Gradient Boosting was slightly slower but performed comparably to XGBoost.

Final Model Recommendation

Based on cross-validation results, XGBoost is the preferred model due to its strong performance in all evaluation metrics.



```
# selected features
X = df[['total_protein', 'total_keratine', 'vitamin', 'manganese', 'iron', 'calcium', 'liver_data']]
y = df['hair_fall']

X_train, X_temp, y_train, y_temp = train_test_split(X, y, test_size=0.3, random_state=42)
X_val, X_test, y_val, y_test = train_test_split(X_temp, y_temp, test_size=0.5, random_state=42)

print(f"Training Set: {X_train.shape[0]} samples")
print(f"Validation Set: {X_val.shape[0]} samples")
print(f"Test Set: {X_test.shape[0]} samples")
```

```
Training Set: 70000 samples
Validation Set: 15000 samples
Test Set: 15000 samples
```

Fig10: Best Model's Evaluation

Feature Importance

The given bar chart illustrates the **Permutation Feature Importance**, which measures the impact of each feature on the model's predictive performance. Permutation importance works by randomly shuffling the values of a feature and evaluating how much the model's accuracy decreases. A higher importance score indicates that the feature significantly contributes to the model's predictions. The feature importance analysis highlights **calcium** as the most critical factor affecting hair fall, followed by **liver health and iron levels**.

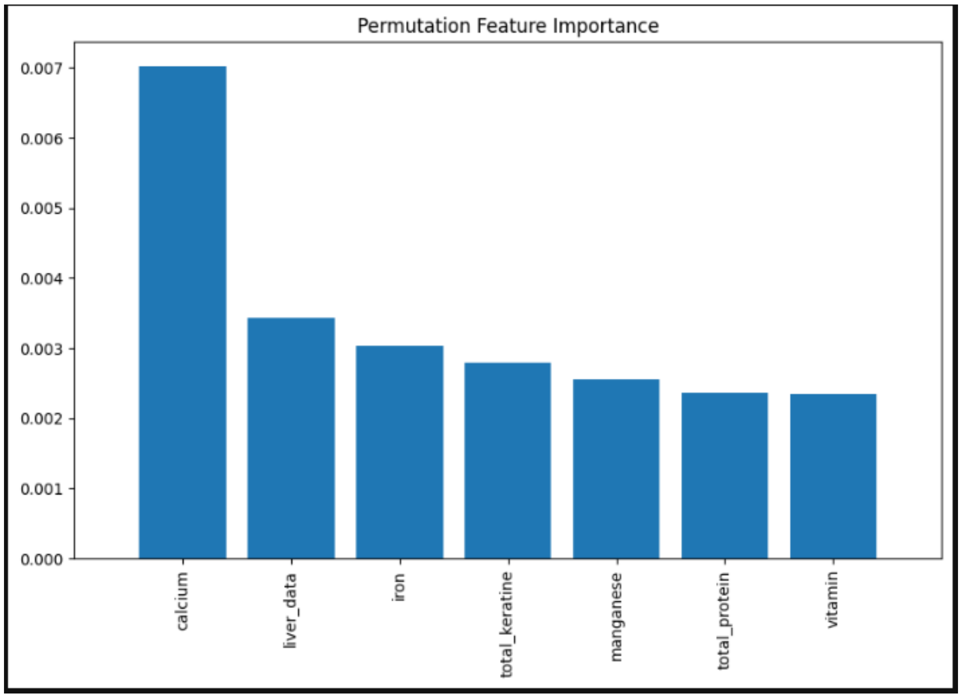
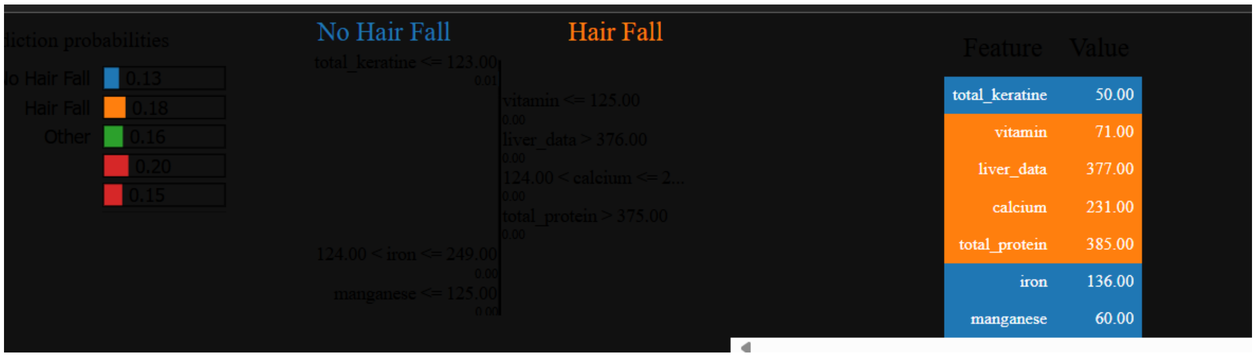


Fig11: Permutation Feature Importance

Lime Implementation



Feature	Value
total_keratine	50.00
vitamin	71.00
liver_data	377.00
calcium	231.00
total_protein	385.00
iron	136.00
manganese	60.00

Fig12: Lime Implementation

Future Considerations

- Hyperparameter tuning (e.g., GridSearch, Bayesian Optimization) for top-performing models.
- Feature engineering improvements, such as interaction terms or domain-specific transformations.
- Ensemble methods, combining multiple models for improved generalization.

Conclusion

This analysis has demonstrated the importance of model selection, cross-validation, and performance evaluation. Based on the findings, XGBoost stands out as the most effective model for this dataset. Further improvements can be explored through hyperparameter tuning and ensemble learning strategies.