



**VIT<sup>®</sup>**  
**B H O P A L**  
[www.vitbhopal.ac.in](http://www.vitbhopal.ac.in)

CHALLENGING TASK

CS3005 - Foundations of Data Science

# Predicting Disease Inheritance in Families

---

GROUP – 26

PRESENTED TO – PROF.GUNJAN ANSARI

# Presented by

---

- ❖ Anvitha.N - 22BSA10128
- ❖ Saksham Gupta - 22BSA10165
- ❖ Anant Bansal - 22BSA10283
- ❖ Mohd Ameen Husain - 22BSA10209
- ❖ Rahul Kumar Jha - 22BSA10263
- ❖ Nishant Verma - 21BCE10104

# STAGE – 1 Discovery

---

---

**Problem statement:** Understanding the transmission of genetic diseases within family tree.

**Hypothesis:**

- ❑ H1: Genetic relationship features (like parental infection status, sibling disease history) will significantly correlate with individual disease susceptibility.
- ❑ H2: Age and genetic diversity scores will show a statistically significant correlation with the probability of disease inheritance.
- ❑ H3: Family structural features (like total siblings, parental characteristics) will contribute meaningfully to predictive model performance.

# STAGE – 2 DATA PREPARATION

---

USING – GOOGLE COLAB AS SANDBOX

A solid orange horizontal bar at the bottom of the slide.

# Dataset creation

---

Creating random names-

1. First we import Random to colab.
2. Then we generate random names using “generate\_name”..

```
# Function to generate random name
def generate_name(gender='M', father_name=None):
    first_names_male = ['Saksham', 'Vivan', 'Aditya', 'Rahul']
    first_names_female = ['Kirti', 'Saanvi', 'Priya', 'Ananya']
    last_names = ['Sharma', 'Patel', 'Bansal', 'Singh', 'Gupta', 'Jha']

    # Select first name based on gender
    first_name = random.choice(first_names_male if gender == 'M' else first_names_female)

    # If father_name is provided, inherit the surname from the father
    last_name = father_name.split()[-1] if father_name else random.choice(last_names)

    return f"{first_name} {last_name}"

# Create a list to store family data
family_data = []
```

---

Creating hierarchies:

1. Generating parent node.

```
# Generation 1: Parents
father1_name = generate_name('M')
father2_name = generate_name('M')
mother1_name = generate_name('F')
mother2_name = generate_name('F')

# Add both parents (Generation 1)
family_data.append([1, father1_name, None, None]) # Father 1
family_data.append([2, mother1_name, None, None]) # Mother 1
family_data.append([3, father2_name, None, None]) # Father 2
family_data.append([4, mother2_name, None, None]) # Mother 2
```

2. Creating children Node.(constraint- each parent can have atmost 3 child ).
3. Repeating this process 4 more time.

```
# List to track children of Generation 1
children_data_gen1 = []

# Maximum number of children per parent
max_children = 3
children_count = {1: 0, 2: 0, 3: 0, 4: 0} # Track number of children for each parent

# Generation 1 children
for i in range(5, 11): # Range is atmost 6 - constraint
    father_id = random.choice([1, 3]) # Randomly choose between father1 and father2
    mother_id = random.choice([2, 4]) # Randomly choose between mother1 and mother2

    # Ensure that no parent has more than 3 children
    if children_count[father_id] < max_children and children_count[mother_id] < max_children:
        #Find the father's name in family_data where the ID matches father_id
        father_name = next(name for fid, name, _, _ in family_data if fid == father_id)

        # i is even the new name will be generated as a male's name // if i is odd, a female's name will be generated
        name = generate_name('M' if i % 2 == 0 else 'F', father_name) # Name based on gender and father's surname

        family_data.append([i, name, father_id, mother_id])
        children_data_gen1.append(i)
        children_count[father_id] += 1
        children_count[mother_id] += 1
```



---

4. Storing all data in a .csv file And previewing our .csv file .

```
print(f"Total number of entries: {len(family_data)}")

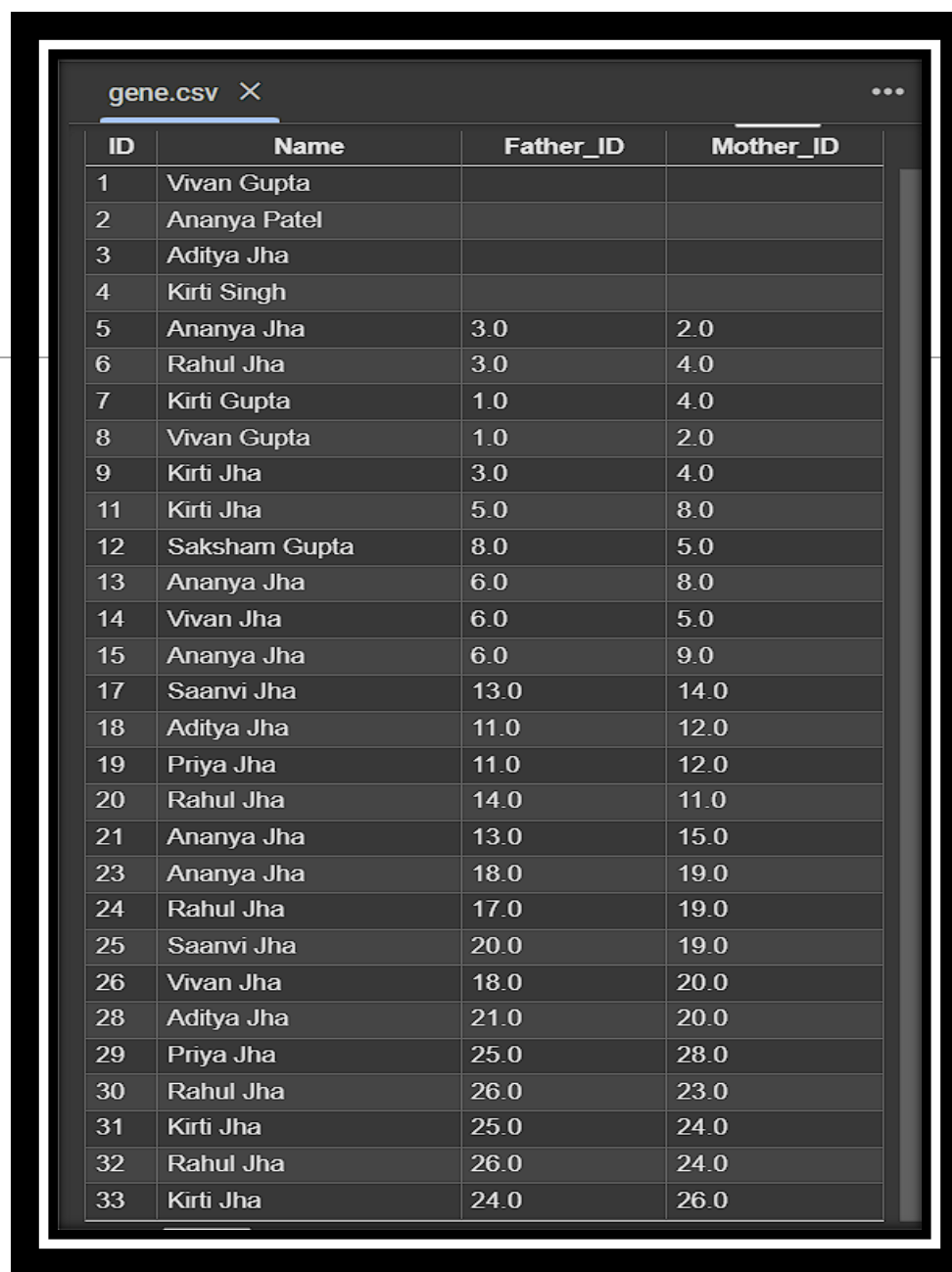
# Create DataFrame
df = pd.DataFrame(family_data, columns=["ID", "Name", "Father_ID", "Mother_ID"])

# Save the file to CSV
df.to_csv('gene.csv', index=False)

# Display the first few rows to confirm
print(df.head())
```

# Output-

Here the data has low granularity



ID	Name	Father_ID	Mother_ID
1	Vivan Gupta		
2	Ananya Patel		
3	Aditya Jha		
4	Kirti Singh		
5	Ananya Jha	3.0	2.0
6	Rahul Jha	3.0	4.0
7	Kirti Gupta	1.0	4.0
8	Vivan Gupta	1.0	2.0
9	Kirti Jha	3.0	4.0
11	Kirti Jha	5.0	8.0
12	Saksham Gupta	8.0	5.0
13	Ananya Jha	6.0	8.0
14	Vivan Jha	6.0	5.0
15	Ananya Jha	6.0	9.0
17	Saanvi Jha	13.0	14.0
18	Aditya Jha	11.0	12.0
19	Priya Jha	11.0	12.0
20	Rahul Jha	14.0	11.0
21	Ananya Jha	13.0	15.0
23	Ananya Jha	18.0	19.0
24	Rahul Jha	17.0	19.0
25	Saanvi Jha	20.0	19.0
26	Vivan Jha	18.0	20.0
28	Aditya Jha	21.0	20.0
29	Priya Jha	25.0	28.0
30	Rahul Jha	26.0	23.0
31	Kirti Jha	25.0	24.0
32	Rahul Jha	26.0	24.0
33	Kirti Jha	24.0	26.0

*Preview*

# STAGE 3 – Model Planning

---

# Plotting a directed graph

1. First import pygraphviz as pgv.(tool for creating, rendering, and visualizing graphs)

ADDING NODE

```
# Load the family tree data from CSV file
df = pd.read_csv('gene.csv')

# Create a directed graph for the family tree
G = nx.DiGraph()

# Add nodes and edges from the DataFrame
for _, row in df.iterrows():
    # used _ because there is no index
    person_id = row['ID']
    name = row['Name']
    father_id = row['Father_ID']
    mother_id = row['Mother_ID']

    # Add the node
    G.add_node(person_id, label=name)

    # Add edges for father and mother
    # .notna return true if the value is not NaN
    if pd.notna(father_id): # Only add the edge if father_id is not NaN
        G.add_edge(father_id, person_id)

    if pd.notna(mother_id): # Only add the edge if mother_id is not NaN
        G.add_edge(mother_id, person_id)
```

---

## 2. Making a layout for the graph.

```
# Define a tree layout for the graph
# parents at the top, children at the bottom
pos = nx.nx_agraph.graphviz_layout(G, prog="dot") # "dot" creates a hierarchical layout
#nx.nx_agraph.graphviz_layout(): is used to compute the layout for a graph G

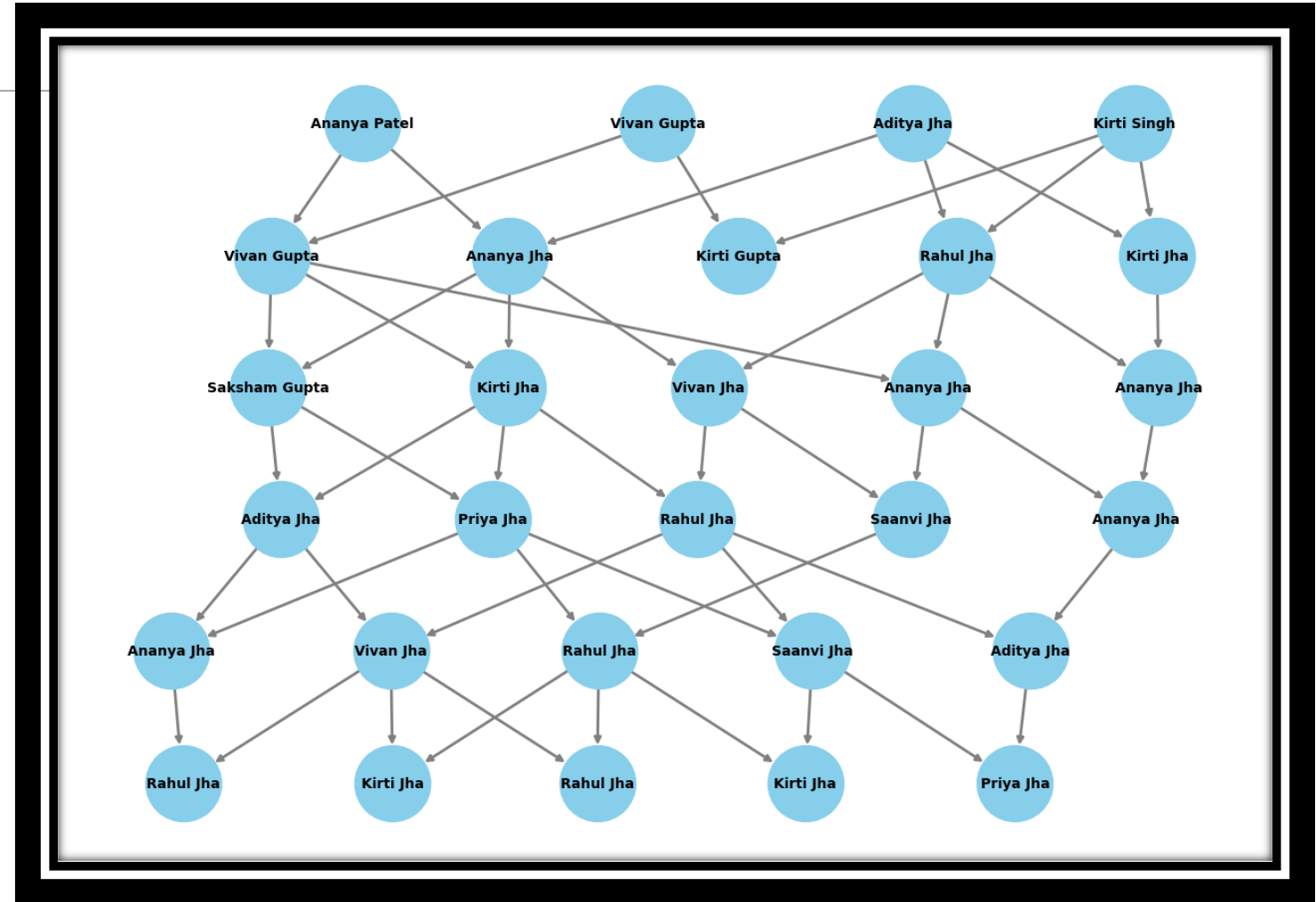
# Plot the family tree
plt.figure(figsize=(12, 8))
# Giving color and design
nx.draw(G, pos, with_labels=True, labels=nx.get_node_attributes(G, 'label'),
        node_size=3000, node_color="skyblue", font_size=10, font_weight='bold',
        width=2, edge_color="gray")

# Show the plot
plt.show()
```

# Output

**Random Forest** is a good choice for analyzing the given network diagram due to its ability to handle:

- Complex relationships: It can capture intricate connections between nodes.
- Feature importance: It identifies key nodes or relationships influencing the network.
- Noise and outliers: *It is robust to data imperfections.*
- Interpretability: It provides insights into the decision-making process.



# STAGE – 4 Machine learning Algorithm

---

RANDOM FOREST



## **1. Step 1:Initialization-**

- The model begins by loading family tree data from a CSV file and verifying the presence of required columns (ID, Name, Father\_ID, Mother\_ID).
- This is crucial because Random Forest relies on well-structured input data, including relationships between individuals, to create meaningful features for classification.

## **2. Step 2:Data Preprocessing -**

- The preprocessing step involves feature engineering, creating attributes like has\_father, has\_mother, and family size metrics.
- These features improve the predictive power of Random Forest by giving it diverse, relevant data to process. Preprocessing also simulates disease status if it's missing, making the dataset ready for model training.

## **3. Step 3:Simulate Disease Inheritance-**

- In the absence of disease status, the model simulates it using genetic factors like parental disease history and age.
- This mimics the Random Forest approach, where each decision tree considers different factors (like parental influence) to predict an outcome.

## **4. Step 4:Feature Engineering-**

- Feature engineering involves creating additional features, such as sibling count and genetic diversity, to provide more context for predictions.
- Random Forest benefits from these features, as it builds multiple decision trees that consider various attributes to make predictions, improving overall model accuracy.



## **5. Step:5 Training the Random Forest Model-**

- The model uses Random Forest to train a classifier, which builds multiple decision trees based on random subsets of data.
- Random Forest reduces overfitting by averaging the results from various trees.
- It makes predictions on disease status, utilizing all engineered features and assessing their importance in the prediction.

## **6. Step:6 Visualizing the Results-**

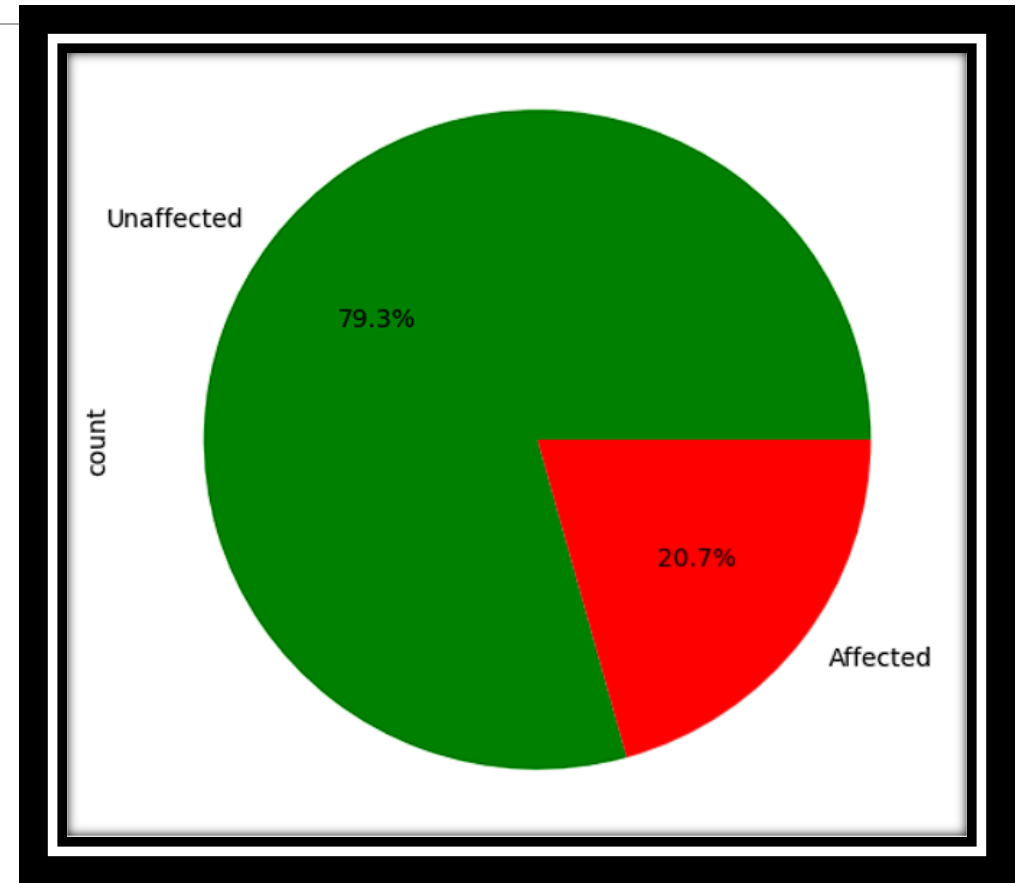
- After training, the model calculates feature importance, showing which features most influence disease prediction.
- Random Forest automatically evaluates feature importance by measuring how much each feature contributes to reducing error in predictions, and this is visualized through a bar chart for better interpretation.

## **7. Step:7 Displaying Results-**

- The model evaluates its performance using metrics such as accuracy, precision, recall, and the confusion matrix.
- Random Forest is robust for classification tasks like this, and these metrics help assess its ability to correctly predict disease status. The confusion matrix, in particular, highlights prediction accuracy across affected and unaffected categories.

# Output

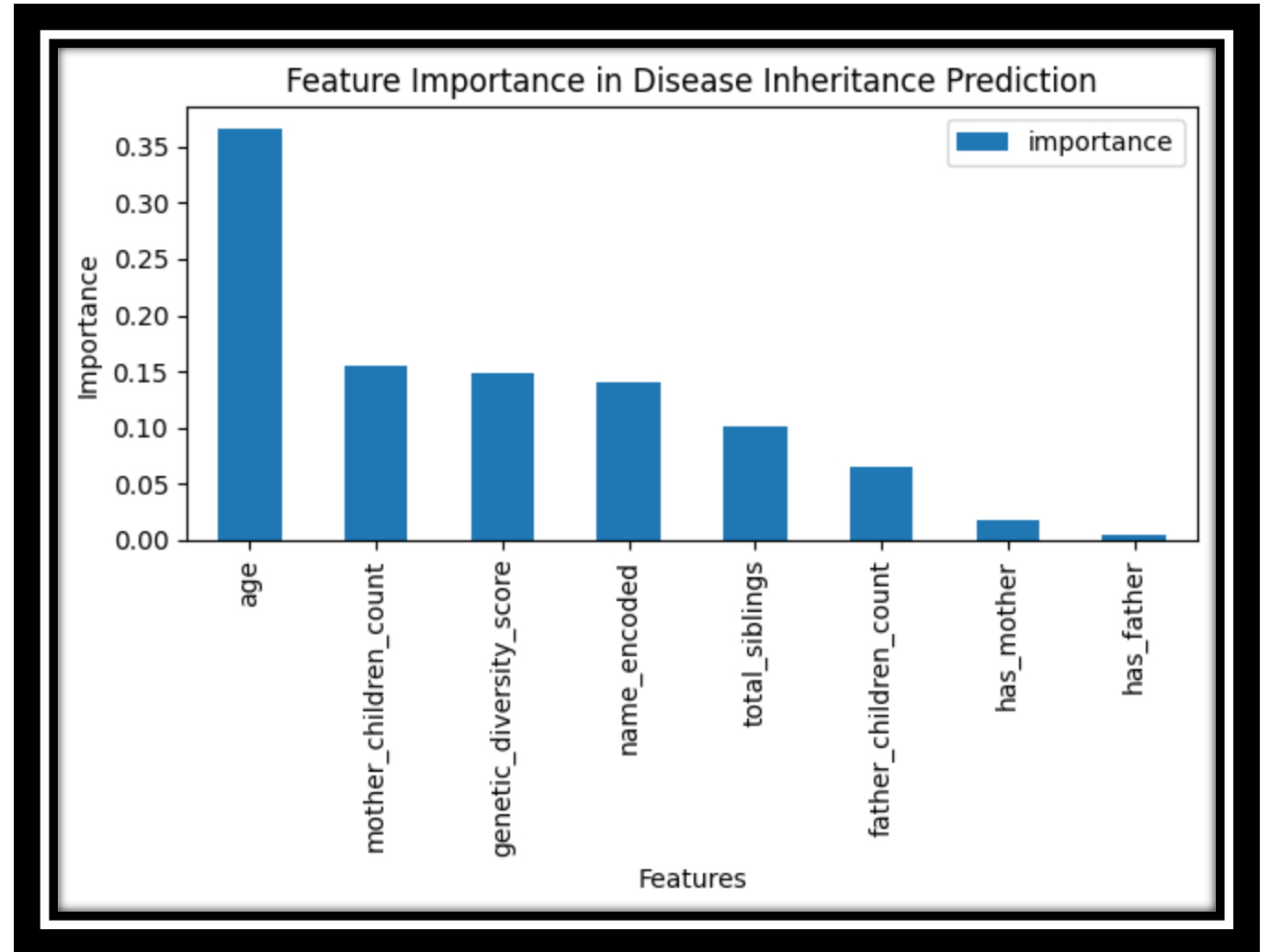
The pie chart shows that 20.7% of the family members are affected by the disease, while 79.3% are not. This suggests that the disease is not very common within this family.



- The bar chart shows the relative importance of different features in predicting disease inheritance.
  - Age is the most important feature, contributing significantly to the prediction.
  - Mother's children count and genetic diversity score are also quite influential.
  - Name encoded, total siblings, and father's children count have moderate importance.
  - Overall, the chart suggests that age, family history, and genetic factors are the most crucial elements in predicting disease inheritance.

Feature Importance:

	feature	importance
5	age	0.366661
3	mother_children_count	0.155828
6	genetic_diversity_score	0.148836
7	name_encoded	0.140623
4	total_siblings	0.101650
2	father_children_count	0.064572
1	has_mother	0.017174
0	has_father	0.004656



- Accuracy: The model is only 50% accurate overall.
- Classification Report:
  - Precision:
    - For class 0 (unaffected), the model is 100% accurate when it predicts this class.
    - However, for class 1 (affected), it's 0% accurate.
  - Recall: The model correctly identifies 50% of the unaffected cases but misses all the affected cases.
  - F1-score: This is a combined metric of precision and recall. It's low for both classes, indicating poor performance.
- Confusion Matrix:
  1. True Positives (TP): 3 (model correctly predicted 3 unaffected individuals).
  2. False Positives (FP): 3 (model incorrectly predicted 3 unaffected individual).
  3. False Negatives (FN): 0 (model missed 0 affected individual) .
  4. True Negatives (TN): 0 (model correctly predicted 0 affected individuals).

#### Random Forest Results:

Accuracy: 0.5

#### Classification Report:

	precision	recall	f1-score	support
0	1.00	0.50	0.67	6
1	0.00	0.00	0.00	0
accuracy			0.50	6
macro avg	0.50	0.25	0.33	6
weighted avg	1.00	0.50	0.67	6

#### Confusion Matrix:

```
[[3 3]
 [0 0]]
```

# Possible explanation for poor performance

---

- ❑ Data Quality: The data used to train the model might be incomplete.
- ❑ Feature Importance: The model might not be using the most relevant features to make predictions.
- ❑ Model Complexity: The Random Forest model might be too complex or too simple for this problem.

❖ Solution: Data Cleaning-Ensure data accuracy and completeness.

# STAGE-5 CORRELATION

---

$$R = (N\Sigma XY - \Sigma X \Sigma Y) / \text{SQRT}[(N\Sigma X^2 - (\Sigma X)^2)(N\Sigma Y^2 - (\Sigma Y)^2)]$$

# STEPS FOR CALCULATING R-Score

## ❑ 1. Selecting Numeric Columns for Correlation:

We use ``select_dtypes(include=[np.number])`` to filter out only numeric columns in the dataset. This is important because the correlation calculation can only be performed on numeric data types, excluding any string-based or categorical columns like names.

## ❑ 2. Encoding Categorical Data (`Name`):

The ``Name`` column is encoded into numeric form using ``.astype('category').cat.codes``, turning each unique name into a distinct numeric value. This allows categorical data to be used in numeric calculations like correlation, even though it's originally text.

## ❑ 3. Computing the Correlation Matrix:

Once we have the numeric columns, the ``.corr()`` method is applied to compute the correlation matrix, which shows how each feature is related to others, including the target variable, ``disease_status``. The values range from -1 to 1, indicating the strength and direction of the relationships.

## ❑ 4. Printing the Correlation results:

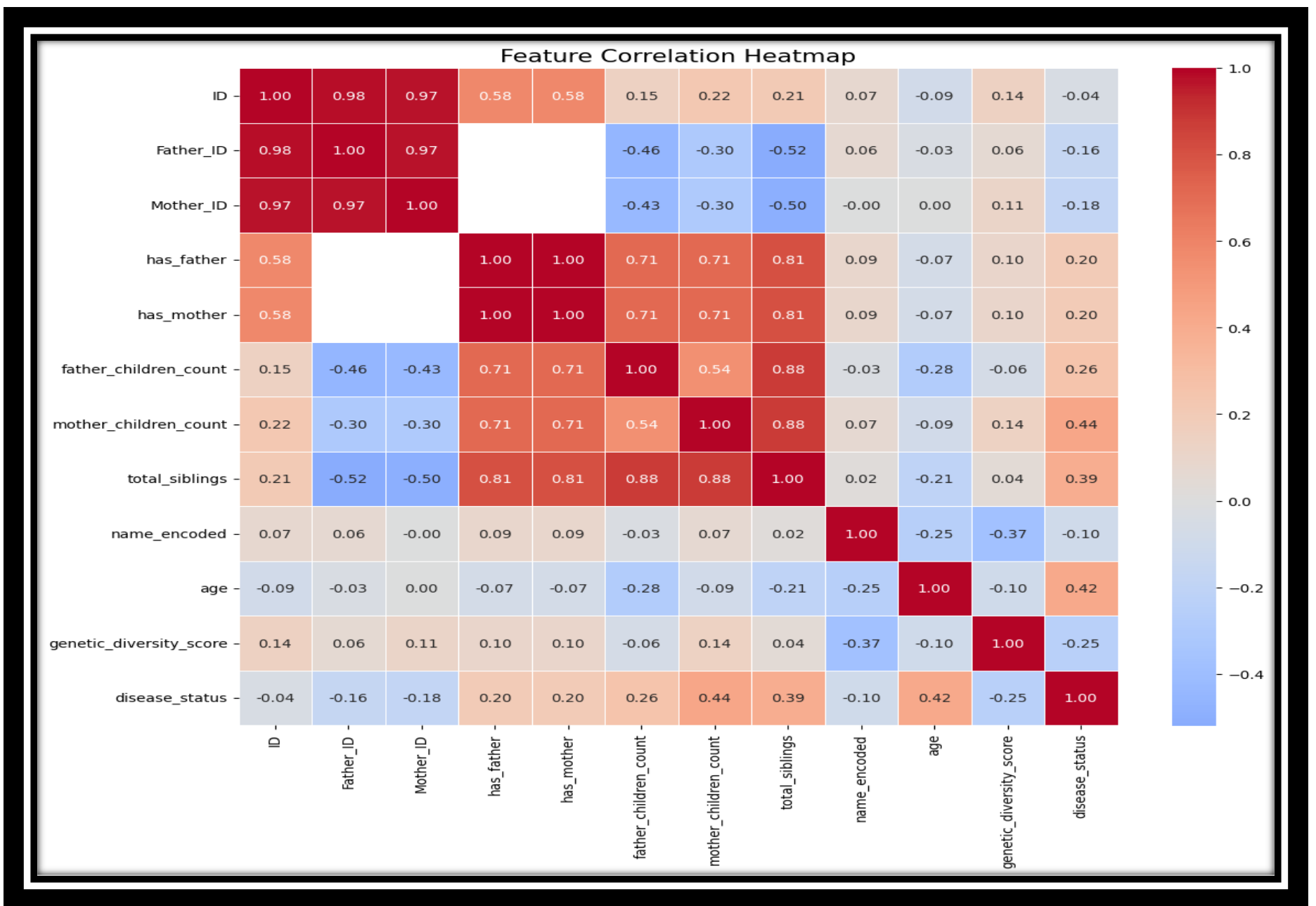
After calculating the full correlation matrix, we specifically extract and display the correlation values for ``disease_status`` with other features. This helps identify which factors are most strongly related to the likelihood of inheriting the disease.

## ❑ 5. Visualizing the Correlation:

The correlation matrix is visualized as a heatmap using Seaborn's ``heatmap`` function. This makes it easy to see the strength and direction of correlations between features at a glance. Strong positive or negative correlations are highlighted with different colors, providing clear insights into feature relationships.

# OUTPUT

Here *Red* is  
*positive*  
*correlation* and  
*blue* *negative*  
*correlation*





# STAGE-6 PREDICTING RESULTS

---

USING PROBABILITY

A solid orange horizontal bar at the bottom of the slide.

# Procedure

---

- ❑ **Data Loading** : The first step of the model involves loading the family genetic data from a CSV file. It then checks if the required columns are present in the dataset: 'ID', 'Name', 'Father\_ID', and 'Mother\_ID'.
- ❑ **Preprocessing Data**: In this step, the model ensures that the key columns (ID, Father\_ID, and Mother\_ID) are converted to numeric values for consistency and to handle any potential inconsistencies in the data. The model also removes any rows where the 'ID' value is missing. This ensures that only valid data is processed for family relationships and genetic modeling.
- ❑ **Feature Engineering**: Feature engineering creates new variables that enhance the predictive capabilities of the model. The model generates binary flags indicating whether a person has a father or mother. It also computes the number of siblings an individual has by counting the children of both parents. Additionally, each individual is assigned a random 'age' between 18 and 80 and a random 'genetic diversity score' to simulate genetic variability within the family.

❑ **Creating Graph:** The family relationships are represented as a directed graph, where each individual is a node. Parent-child relationships are represented by directed edges between nodes, with parents pointing to their children. This graph structure allows for the analysis of inheritance patterns and the calculation of disease probabilities based on family relationships.

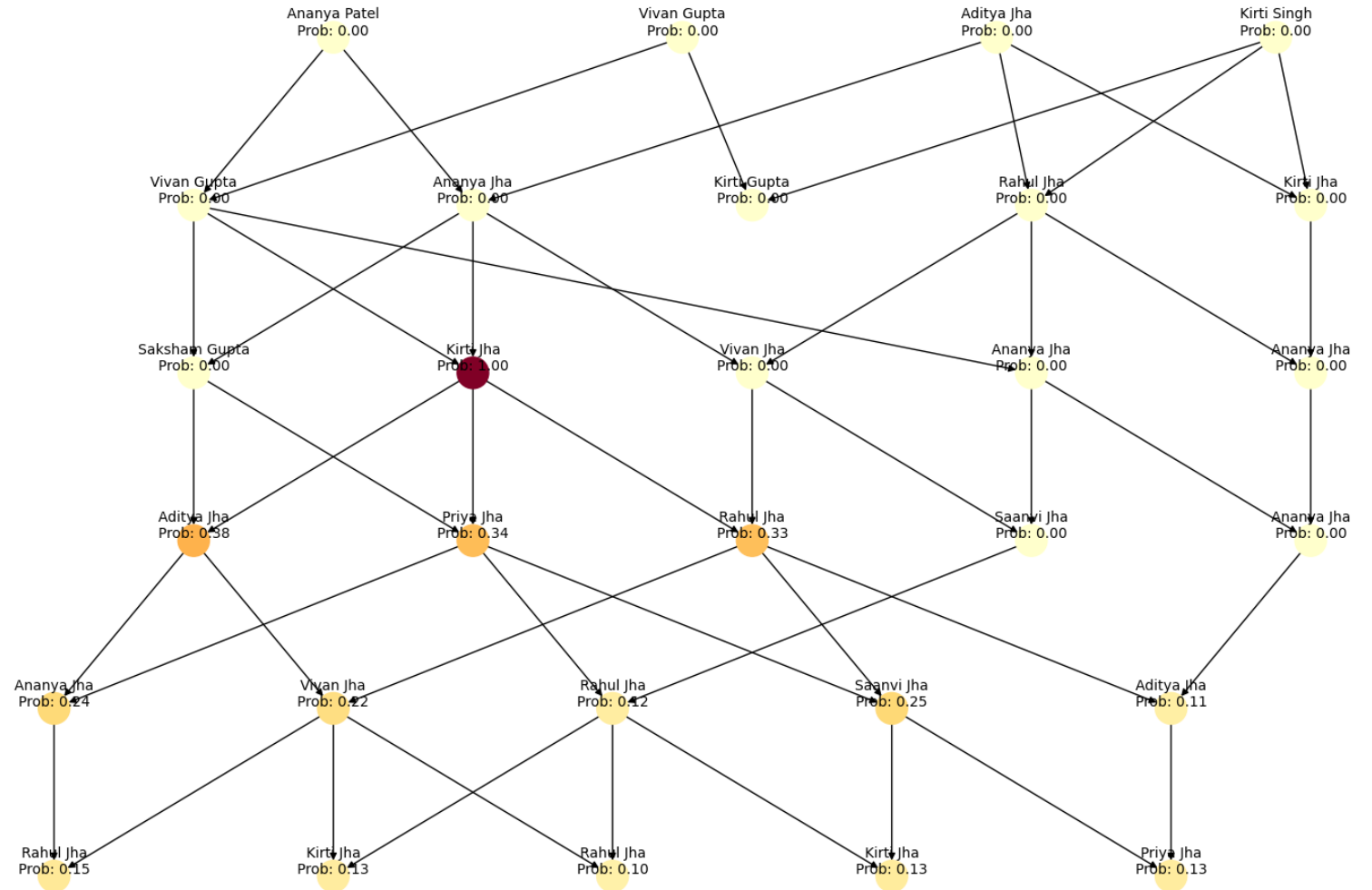
❑ **Calculating Inheritance Probability:** The model calculates the probability of inheriting a genetic disease by considering both the disease probabilities of an individual's parents and their own genetic features. The initial infection status of specific family members is set, and their inheritance probabilities are marked as 1. For other individuals, the model computes the probability based on factors such as age, number of siblings, and parental influence. *Feature normalization* ensures that each variable contributes proportionally to the final probability.

❑ **Generating Inheritance Report:** After calculating the inheritance probabilities for all family members, the model generates a report. This report lists each individual's probability of inheriting the disease, sorted from the highest to lowest probability. The report helps to identify individuals who are at higher risk, offering valuable insights into how the disease might spread through the family.

❑ **Visualizing the Graph:** The final step is the visualization of the family tree using a hierarchical layout. The model uses PyGraphviz to position nodes in a clear, top-down manner, ensuring the tree structure is easily understandable. Nodes are color-coded based on the inheritance probability, with warmer colors representing higher probabilities. Each node is labeled with the individual's name and their inheritance probability, providing a visual representation of the genetic inheritance process within the family.

# OUTPUT

	Inheritance Probability	Name
11	1.000000	Kirti Jha
18	0.376914	Aditya Jha
19	0.343284	Priya Jha
20	0.333702	Rahul Jha
25	0.250983	Saanvi Jha
23	0.244677	Ananya Jha
26	0.215582	Vivan Jha
30	0.151204	Rahul Jha
31	0.132919	Kirti Jha
33	0.130400	Kirti Jha
29	0.130041	Priya Jha
24	0.121389	Rahul Jha
28	0.114254	Aditya Jha
32	0.100960	Rahul Jha
7	0.000000	Kirti Gupta
8	0.000000	Vivan Gupta
5	0.000000	Ananya Jha
6	0.000000	Rahul Jha
17	0.000000	Saanvi Jha
15	0.000000	Ananya Jha
14	0.000000	Vivan Jha
13	0.000000	Ananya Jha
12	0.000000	Saksham Gupta
9	0.000000	Kirti Jha
21	0.000000	Ananya Jha



# Inference

---

- ❑ H1=Hypothesis 1: Genetic relationship features will significantly correlate with individual disease susceptibility.
- ❑ *Father\_ID* and *Mother\_ID*: These have a strong positive correlation with each other and a moderate negative correlation with disease status, suggesting that specific parental lineages might be associated with a lower risk of the disease.
- ❑ *has\_father* and *has\_mother*: These have a strong positive correlation, indicating that having both parents might be associated with a slightly higher risk.

❖ Hence H1 is true

# Inference

---

- ❑ H2=Hypothesis 2: Age and genetic diversity scores will show a statistically significant correlation with the probability of disease inheritance.
- ❑ *Age*: Has a moderate positive correlation with disease status, suggesting that older individuals might be at a higher risk.
- ❑ *genetic\_diversity\_score*: Has a weak negative correlation, indicating that higher genetic diversity might be associated with a slightly lower risk.

❖ Hence H2 is true

# Inference

---

- ❑ H3=Hypothesis 3: Family structural features will contribute meaningfully to predictive model performance.
- ❑ *total\_siblings*: Has a strong positive correlation with disease status, suggesting that having more siblings might be associated with a higher risk.
- ❑ *father\_children\_count* and *mother\_children\_count*: These also have strong positive correlations with disease status, indicating that having more children might be associated with a higher risk.

❖ Hence H3 is true

# POTENTIAL FUTURE APPLICATIONS

---

- ❖ Agricultural Breeding
- ❖ Wildlife Conservation
- ❖ Medical Disease Prevention
- ❖ Environmental Adaptation
- ❖ Pharmaceutical Research



# Reference

---

- [Machine Learning for Plant Breeding and Biotechnology](#) (Reviewed by – Saksham Gupta:22BSA10165)
- <https://pmc.ncbi.nlm.nih.gov/articles/PMC11141855/> (Reviewed by – Rahul Jha:22BSA10263)
- [Machine learning-based genetic diagnosis models for hereditary hearing loss by the GJB2, SLC26A4 and MT-RNR1 variants – eBioMedicine](#) (Reviewed by - Nishant Verma:21BCE10104)
- <https://jesit.springeropen.com/articles/10.1186/s43067-023-00108-y> (Reviewed by - Anant Bansal:22BSA10283)
- <https://ieeexplore.ieee.org/abstract/document/7912315> (Reviewed by - Anvitha.N:22BSA10128)

❖ [GOOGLE COLAB](#)

THANK YOU !

---