DNA Trait Predictor

Agile Development Plan (Scrum Framework)

Project Overview

Project Name: DNA Trait Predictor

Duration: 4 weeks (March 10 – April 7, 2026)

Sprint Length: 1 week per sprint

Goal: Build an AI-powered application that predicts eye color, hair color, and ancestry from DNA data (SNPs) using machine learning, complete with a user-friendly GUI.

Tech Stack

- **Python 3.8+**: Core language
- **pandas**: CSV data processing and feature extraction
- **scikit-learn**: Machine learning (Random Forest classifiers)
- **numpy**: Numerical computations
- **tkinter**: GUI framework (built-in)
- **matplotlib**: Confidence bar visualization
- **pickle**: Model serialization

Product Backlog

The product backlog contains 16 user stories distributed across 4 sprints. Total story points: 46.

ID	User Story	Sprint	Priority	Points
US-01	As a user, I want to load SNP data from a CSV file so that I can analyze genetic markers	1	High	**3**
US-02	As a user, I want to filter SNPs by chromosome and position so that I can extract relevant markers	1	High	**3**

US-03	As a user, I want to visualize SNP distributions so that I can understand my dataset	1	Medium	**2**
US-04	As a developer, I want to create a training dataset for eye color so that I can train the model	1	High	**4**
US-05	As a user, I want to train a Random Forest classifier for eye color so that I can predict blue/green/hazel/brown eyes	2	High	**5**
US-06	As a developer, I want to split data into train/test sets so that I can evaluate model accuracy	2	High	**2**
US-07	As a user, I want to evaluate the eye color model with accuracy metrics so that I know how well it works	2	High	**3**
US-08	As a developer, I want to save the trained model to disk so that I can reuse it without retraining	2	High	**2**
US-09	As a user, I want to make predictions on new SNP data so that I can see eye color predictions	2	High	**2**
US-10	As a user, I want to train a hair color classifier so that I can predict black/brown/blonde/red hair	3	High	**4**
US-11	As a user, I want to train an ancestry predictor so that I can estimate continental ancestry percentages	3	High	**5**
US-12	As a developer, I want to create a unified prediction pipeline so that I can run all three models on one dataset	3	Medium	**3**
US-13	As a user, I want to see confidence scores for each prediction so that I know how certain the AI is	3	Medium	**3**
US-14	As a user, I want a tkinter GUI with input fields for SNPs so that I can easily input genetic data	4	High	**4**
US-15	As a user, I want to upload a CSV file through the GUI so that I can analyze bulk data	4	Medium	**2**

US-16	As a user, I want to see trait predictions displayed with confidence bars so that results are clear and visual	4	High	**3**

Sprint 1: Data Pipeline (March 10-17, 2026)

Goal: Build the foundation—load, parse, and explore SNP data from CSV files.

Sprint 1 User Stories & Tasks

ID	Tasks	Acceptance Criteria	Points
US-01		Script loads CSV and displays first 10 rows with correct columns	**3**
US-02		User can filter for specific SNPs (e.g., rs12913832) and get accurate results	**3**
US-03		Script generates 3 plots showing data quality and distribution	**2**
US-04		CSV file with 6 SNP columns + eye_color label, 80/20 train/test split	**4**

Sprint 1 Deliverables

- data_loader.py — CSV parser module
- snp_filter.py — Filtering utilities
- visualize.py — Data exploration plots
- eye_color_train.csv / eye_color_test.csv — Labeled datasets

Sprint 2: Eye Color ML Model (March 18-25, 2026)

Goal: Train, evaluate, and deploy the first machine learning model (eye color predictor).

Sprint 2 User Stories & Tasks

ID	Tasks	Acceptance Criteria	Points

US-05		Model trains without errors and produces prediction probabilities for 4 classes	**5**
US-06		Train/test split maintains class distribution (±2%)	**2**
US-07		Model achieves >75% test accuracy with confusion matrix visualization	**3**
US-08		Saved model can be loaded and produces identical predictions	**2**
US-09		Given new SNP data, model returns eye color with confidence percentages	**2**

Sprint 2 Deliverables

- train_eye_model.py — Training script
- models/eye_color_rf.pkl — Serialized Random Forest model
- predict_eye_color.py — Prediction interface
- evaluation_report.txt — Accuracy metrics and confusion matrix

Sprint 3: Multi-Trait Models (March 26 – April 2, 2026)

Goal: Train hair color and ancestry models, then integrate all three into a unified prediction pipeline.

Sprint 3 User Stories & Tasks

ID	Tasks	Acceptance Criteria	Points
US-10		Hair color model achieves >70% test accuracy and saves to hair_color_rf.pkl	**4**
US-11		Ancestry model predicts continental percentages (e.g., 85% EUR, 15% EAS) with >80% accuracy	**5**

US-12		Single function call returns all predictions from one SNP dataset	**3**
US-13		Predictions include confidence (e.g., 'Brown eyes: 92% confidence')	**3**

Sprint 3 Deliverables

- train_hair_model.py — Hair color training script
- train_ancestry_model.py — Ancestry training script
- models/hair_color_rf.pkl — Hair color model
- models/ancestry_rf.pkl — Ancestry model
- pipeline.py — Unified prediction interface

Sprint 4: GUI & Deployment (April 3-7, 2026)

Goal: Build a tkinter GUI that lets users input SNPs and see predictions visually.

Sprint 4 User Stories & Tasks

ID	Tasks	Acceptance Criteria	Points
US-14		GUI launches, accepts input for 6 SNPs, and displays predictions when button clicked	**4**
US-15		User can upload CSV and see predictions for multiple individuals	**2**
US-16		Results display with horizontal bar charts showing confidence percentages	**3**

Sprint 4 Deliverables

- app.py — Main GUI application (executable)
- README.md — Installation and usage instructions

- requirements.txt — Python dependencies

Machine Learning Model Details

Model Architecture: Random Forest Classifier

Parameter	Value	Rationale
n_estimators	**100**	Balance between accuracy and training time. 100 trees provide stable predictions without overfitting.
max_depth	**10**	Prevents overfitting on small genetic datasets while allowing enough complexity to capture SNP interactions.
min_samples_split	**4**	Avoids creating leaf nodes from noise. Ensures each split represents meaningful genetic patterns.
min_samples_leaf	**2**	Minimum samples per leaf. Reduces variance and improves generalization.
criterion	**gini**	Gini impurity for classification. Standard for categorical traits (eye/hair color).
class_weight	**balanced**	Addresses class imbalance in training data (e.g., more brown eyes than green).
random_state	**42**	Reproducibility. Same seed ensures consistent results across runs.

Feature Engineering

Input Features Per Trait:

- **Eye Color:** 6 SNPs (rs12913832, rs1800407, rs12896399, rs1393350, rs12203592, rs1667394)
- **Hair Color:** 8 SNPs from MC1R, TYR, TYRP1, KITLG genes
- **Ancestry:** 50+ Ancestry Informative Markers (AIMs) from 1000 Genomes

Encoding:
Genotypes are encoded as integers: AA=0, AG=1 (or AC, AT, etc.), GG=2 (or CC, TT). This preserves the additive genetic model (0 = homozygous reference, 1 = heterozygous, 2 = homozygous alternate).

Sprint Calendar

Sprint	Dates	Focus	Story Points	Velocity
Sprint 1	March 10-17	Data Pipeline	**12**	[TBD]
Sprint 2	March 18-25	Eye Color ML Model	**14**	[TBD]
Sprint 3	March 26 – Apr 2	Multi-Trait Models	**10**	[TBD]
Sprint 4	April 3-7	GUI & Deployment	**10**	[TBD]

Note: Velocity will be calculated after each sprint by tracking completed story points. Target velocity: 10-12 points per week.

Risk Register

Risk	Impact	Mitigation	Priority
Insufficient training data	Models perform poorly (<60% accuracy)	Use simulated data + oversample minority classes	**High**
Students lack Python skills	Cannot complete coding tasks	Provide starter code templates + live coding demos	**Medium**

Model overfitting	High train accuracy, low test accuracy	Use cross-validation + hyperparameter tuning	Medium
GUI complexity	tkinter implementation takes too long	Use simplified layout + pair programming	Low
Dataset privacy concerns	Students uncomfortable using real genetic data	Use only anonymized public datasets (OpenSNP) + simulated data	High

Definition of Done

A user story is considered 'Done' when:

☑ Code is written and tested (unit tests where applicable)

☑ Acceptance criteria are met and verified

☑ Code is committed to GitHub with descriptive commit message

☑ Documentation is updated (comments, README, or docstrings)

☑ Demo/walkthrough completed in weekly session

Project Success Metrics

- **Model Accuracy:** All three models achieve >70% test accuracy
- **GUI Functionality:** Users can input SNPs and see predictions without errors
- **Student Engagement:** ≥75% of students complete all 4 weeks
- **Code Quality:** Final project runs on fresh Python environment with only requirements.txt dependencies
- **Deliverable:** Working .py application that can be demonstrated to non-technical audience