**Objective**  
Work in groups to solve, analyze, or visualize a problem using machine learning (ML) with other technologies learned.**Requirements**

1. Problem Statement: Find a problem worth solving, analyzing, or visualizing.
2. Machine Learning: Use ML with Scikit-learn and/or another ML library.
3. Dataset: Use a dataset with at least 100 records.
4. Technologies: Use at least two of the following:
   * Python Pandas
   * Python Matplotlib
   * HTML/CSS/Bootstrap
   * JavaScript Plotly
   * JavaScript Leaflet
   * SQL Database
   * MongoDB Database
   * Google Cloud SQL
   * Amazon AWS
   * Tableau

**Project Ideas**

* Finance: credit score analysis, chatbots, consumer spending trends
* Healthcare: disease symptom recognition, image classification, patient history analysis
* Custom: choose an industry and apply ML to a problem

**Guidelines**

1. Collaboration: Work closely with your group, communicate regularly, and use tools like GitHub Projects.
2. Data Sources: Use curated sources like data.world, Kaggle, (link unavailable), and more.
3. Data Cleanup and Analysis: Explore, clean, and analyze data using Jupyter notebooks.

**Deliverables**

1. GitHub Repository: Submit the URL on presentation day.
2. Presentation: Rehearse and present your project.

**Resources**

1. Instructional Team: Support during classes and office hours.
2. Learning Assistants and Tutors: Help with topics as needed.

Project 4 and Project 4 Presentations are due on Thursday March 13th

white_check_markeyesraised_hands

Message 01-live

I'll help you plan and structure a project for zoo animal classification using machine learning. This is a great choice for a group project as it involves classification (a core ML task) and has a clearly defined dataset with multiple attributes.

Let's break down what your project could entail:

**Project Plan: Zoo Animal Classification**

**1. Problem Statement**

Your project aims to build a machine learning model that can accurately classify animals into their taxonomic classes based on physical and behavioral attributes. This has applications in biodiversity studies, automated species identification, and educational tools.

**2. Technologies to Implement**

Based on your requirements, I recommend using:

* **Python with Scikit-learn** for machine learning models
* **Python Pandas** for data manipulation and preprocessing
* **Python Matplotlib/Seaborn** for data visualization
* **HTML/CSS/Bootstrap** for creating a web interface (optional)

**3. Dataset Analysis**

The zoo.csv dataset contains 17 animal attributes plus class type, while class.csv provides metadata about the classification scheme. Here's a preliminary breakdown:

* 17 features (mostly boolean, one numeric)
* Target variable: class\_type (7 classes)
* Supplementary information in class.csv for interpretation

**4. Project Workflow**

**Data Exploration and Preprocessing**

1. Load the datasets using Pandas
2. Explore data statistics and distributions
3. Check for missing values or inconsistencies
4. Create visualizations to understand feature relationships
5. Encode categorical variables if needed

**Model Development**

1. Split data into training and testing sets
2. Try multiple classification algorithms:
   * Decision Trees
   * Random Forest
   * Support Vector Machines
   * K-Nearest Neighbors
   * Neural Networks (optional)
3. Evaluate models using metrics like accuracy, precision, recall, and F1-score
4. Perform hyperparameter tuning to optimize performance

**Feature Analysis**

1. Identify the most important features for classification
2. Analyze how different attributes correlate with animal classes
3. Visualize decision boundaries for the models

**Web Interface (Optional)**

1. Develop a simple web app where users can input animal attributes
2. Use the trained model to predict the class in real-time
3. Display confidence scores and related animals

**5. Code Structure (Starter Example)**

Let me provide a basic structure for how you might start this project:

python

Copy

# Import libraries

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

from sklearn.model\_selection import train\_test\_split

from sklearn.preprocessing import StandardScaler

from sklearn.ensemble import RandomForestClassifier

from sklearn.metrics import classification\_report, confusion\_matrix

# Load the datasets

zoo\_df = pd.read\_csv('zoo.csv')

class\_df = pd.read\_csv('class.csv')

# Explore the data

print(zoo\_df.head())

print(zoo\_df.info())

print(zoo\_df.describe())

# Merge datasets for better insights

merged\_df = pd.merge(zoo\_df, class\_df, left\_on='class\_type', right\_on='Class\_Number')

# Visualize data

plt.figure(figsize=(10, 6))

sns.countplot(data=zoo\_df, x='class\_type')

plt.title('Distribution of Animal Classes')

plt.show()

# Prepare features and target

X = zoo\_df.drop(['animal\_name', 'class\_type'], axis=1)

y = zoo\_df['class\_type']

# Split the dataset

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.25, random\_state=42)

# Train a model

rf\_model = RandomForestClassifier(n\_estimators=100, random\_state=42)

rf\_model.fit(X\_train, y\_train)

# Evaluate the model

y\_pred = rf\_model.predict(X\_test)

print(classification\_report(y\_test, y\_pred))

# Visualize confusion matrix

cm = confusion\_matrix(y\_test, y\_pred)

plt.figure(figsize=(10, 8))

sns.heatmap(cm, annot=True, fmt='d', cmap='Blues')

plt.xlabel('Predicted')

plt.ylabel('Actual')

plt.title('Confusion Matrix')

plt.show()

# Feature importance

feature\_importance = pd.DataFrame({

'Feature': X.columns,

'Importance': rf\_model.feature\_importances\_

}).sort\_values('Importance', ascending=False)

plt.figure(figsize=(10, 6))

sns.barplot(data=feature\_importance, x='Importance', y='Feature')

plt.title('Feature Importance')

plt.show()

**6. Advanced Extensions**

If you want to make your project stand out:

1. **Model Comparison Dashboard**: Create visualizations comparing multiple ML algorithms
2. **Interactive Prediction Tool**: Build a tool where users can adjust animal features and see classification change in real-time
3. **Hierarchical Classification**: Implement a two-stage classifier that first identifies broad categories
4. **Animal Similarity Analysis**: Add functionality to find the most similar animals based on attributes

Would you like me to elaborate on any particular aspect of this project plan? For example, I can provide more details on specific visualization techniques, model tuning approaches, or web interface implementation.

I'll help you plan and structure a project for zoo animal classification using machine learning. This is a great choice for a group project as it involves classification (a core ML task) and has a clearly defined dataset with multiple attributes.

Let's break down what your project could entail:

**Project Plan: Zoo Animal Classification**

**1. Problem Statement**

Your project aims to build a machine learning model that can accurately classify animals into their taxonomic classes based on physical and behavioral attributes. This has applications in biodiversity studies, automated species identification, and educational tools.

**2. Technologies to Implement**

Based on your requirements, I recommend using:

* **Python with Scikit-learn** for machine learning models
* **Python Pandas** for data manipulation and preprocessing
* **Python Matplotlib/Seaborn** for data visualization
* **HTML/CSS/Bootstrap** for creating a web interface (optional)

**3. Dataset Analysis**

The zoo.csv dataset contains 17 animal attributes plus class type, while class.csv provides metadata about the classification scheme. Here's a preliminary breakdown:

* 17 features (mostly boolean, one numeric)
* Target variable: class\_type (7 classes)
* Supplementary information in class.csv for interpretation

**4. Project Workflow**

**Data Exploration and Preprocessing**

1. Load the datasets using Pandas
2. Explore data statistics and distributions
3. Check for missing values or inconsistencies
4. Create visualizations to understand feature relationships
5. Encode categorical variables if needed

**Model Development**

1. Split data into training and testing sets
2. Try multiple classification algorithms:
   * Decision Trees
   * Random Forest
   * Support Vector Machines
   * K-Nearest Neighbors
   * Neural Networks (optional)
3. Evaluate models using metrics like accuracy, precision, recall, and F1-score
4. Perform hyperparameter tuning to optimize performance

**Feature Analysis**

1. Identify the most important features for classification
2. Analyze how different attributes correlate with animal classes
3. Visualize decision boundaries for the models

**Web Interface (Optional)**

1. Develop a simple web app where users can input animal attributes
2. Use the trained model to predict the class in real-time
3. Display confidence scores and related animals

**5. Code Structure (Starter Example)**

Let me provide a basic structure for how you might start this project:

python

Copy

# Import libraries

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

from sklearn.model\_selection import train\_test\_split

from sklearn.preprocessing import StandardScaler

from sklearn.ensemble import RandomForestClassifier

from sklearn.metrics import classification\_report, confusion\_matrix

# Load the datasets

zoo\_df = pd.read\_csv('zoo.csv')

class\_df = pd.read\_csv('class.csv')

# Explore the data

print(zoo\_df.head())

print(zoo\_df.info())

print(zoo\_df.describe())

# Merge datasets for better insights

merged\_df = pd.merge(zoo\_df, class\_df, left\_on='class\_type', right\_on='Class\_Number')

# Visualize data

plt.figure(figsize=(10, 6))

sns.countplot(data=zoo\_df, x='class\_type')

plt.title('Distribution of Animal Classes')

plt.show()

# Prepare features and target

X = zoo\_df.drop(['animal\_name', 'class\_type'], axis=1)

y = zoo\_df['class\_type']

# Split the dataset

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.25, random\_state=42)

# Train a model

rf\_model = RandomForestClassifier(n\_estimators=100, random\_state=42)

rf\_model.fit(X\_train, y\_train)

# Evaluate the model

y\_pred = rf\_model.predict(X\_test)

print(classification\_report(y\_test, y\_pred))

# Visualize confusion matrix

cm = confusion\_matrix(y\_test, y\_pred)

plt.figure(figsize=(10, 8))

sns.heatmap(cm, annot=True, fmt='d', cmap='Blues')

plt.xlabel('Predicted')

plt.ylabel('Actual')

plt.title('Confusion Matrix')

plt.show()

# Feature importance

feature\_importance = pd.DataFrame({

'Feature': X.columns,

'Importance': rf\_model.feature\_importances\_

}).sort\_values('Importance', ascending=False)

plt.figure(figsize=(10, 6))

sns.barplot(data=feature\_importance, x='Importance', y='Feature')

plt.title('Feature Importance')

plt.show()

**6. Advanced Extensions**

If you want to make your project stand out:

1. **Model Comparison Dashboard**: Create visualizations comparing multiple ML algorithms
2. **Interactive Prediction Tool**: Build a tool where users can adjust animal features and see classification change in real-time
3. **Hierarchical Classification**: Implement a two-stage classifier that first identifies broad categories
4. **Animal Similarity Analysis**: Add functionality to find the most similar animals based on attributes

Would you like me to elaborate on any particular aspect of this project plan? For example, I can provide more details on specific visualization techniques, model tuning approaches, or web interface implementation.

I'll create a comprehensive example showing how to implement the zoo animal classification project using Python, Pandas, and Matplotlib. This will give you a solid foundation that your group can build upon.

Zoo Animal Classification Project with Python, Pandas, and Matplotlib

Click to open code

Tap to open

This implementation provides a complete workflow for the zoo animal classification project using Python, Pandas, and Matplotlib. Let me walk you through the key components:

**Data Exploration and Visualization**

The code generates several visualizations to help understand the dataset:

1. **Class Distribution** - Shows how many animals belong to each taxonomic class
2. **Correlation Heatmap** - Reveals relationships between different animal features
3. **Boolean Feature Distribution** - Shows how binary features like "hair" or "feathers" are distributed across classes
4. **Legs Distribution** - Visualizes how the number of legs varies by animal class
5. **PCA Visualization** - Uses dimensionality reduction to plot animals in 2D space based on their features

**Machine Learning Models**

The code implements and compares four different classification algorithms:

* Decision Tree
* Random Forest
* K-Nearest Neighbors
* Support Vector Machine

For each model, it:

* Trains on a portion of the data
* Evaluates performance with accuracy metrics and classification reports
* Generates confusion matrices to visualize prediction errors
* Creates a bar chart comparing all models' accuracy scores

**Feature Importance Analysis**

The analysis identifies which animal attributes are most important for classification using the Random Forest model. This helps understand which physical characteristics best determine an animal's taxonomic class.

**Animal Similarity Analysis**

The code includes a function to find animals most similar to a given animal based on their features using cosine similarity. This could be extended into an interesting feature of your application.

**Next Steps for Your Group**

1. **Web Interface**: Create a simple web app using HTML/CSS/Bootstrap where users can:
   * Input animal characteristics
   * Get predictions about the animal's class
   * View similar animals
2. **Model Optimization**: Uncomment the GridSearchCV section to fine-tune the hyperparameters for better performance
3. **Interactive Visualizations**: Convert some static Matplotlib plots to interactive JavaScript visualizations using Plotly
4. **Feature Engineering**: Experiment with creating new features or transforming existing ones
5. **Deployment**: Set up the model to run in a cloud environment like AWS or Google Cloud

The code is well-commented and saves all visualizations as PNG files, making it easy to include them in your presentation. It also saves the trained model for later use in your application.

Would you like me to explain any specific part of the code in more detail or provide guidance on how to extend it for your project?