SnapML: A Visual Programming Tool for Machine Learning

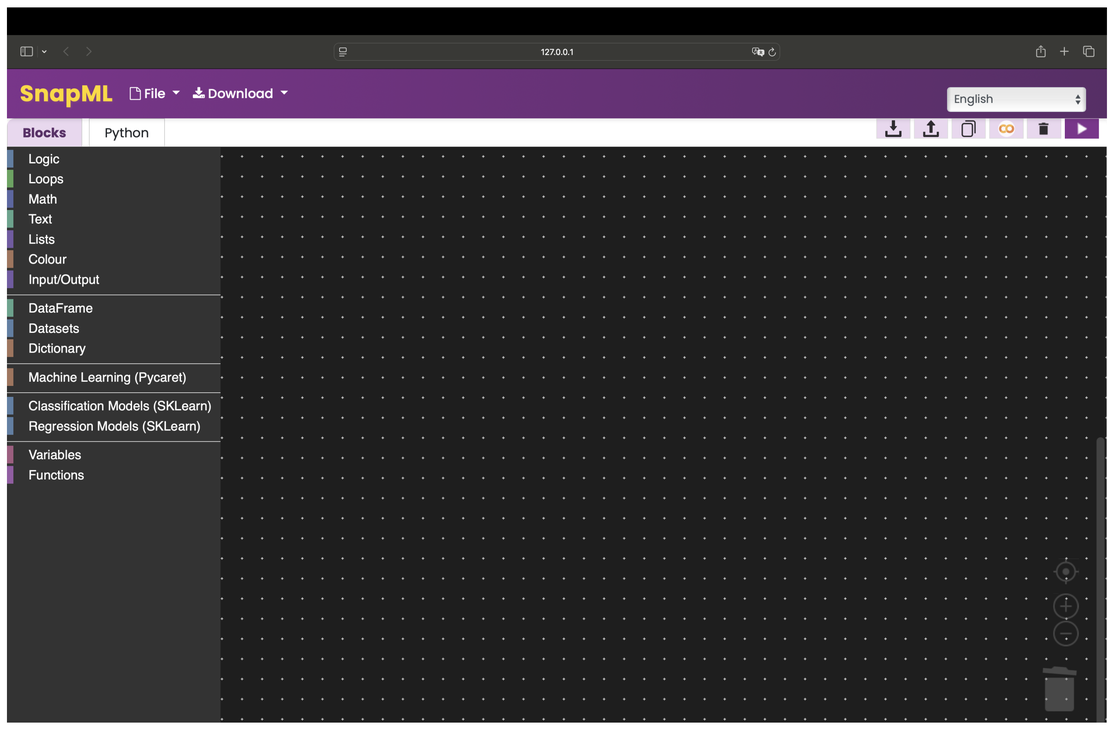
A Block-Based Drag-and-Drop Interface for Machine Learning Development

CS5351 Software Engineering Final Project — Team 8

Team Leader: LI ZHIYU

Members: CHANG CHONGYU, CHEN DATANG, FENG ZIXIAO, LI GUOQIONG, SUN CHENGWEI, YE KAI, ZONG WEIHANG

1 Project Overview and Contributions



SnapML is a drag-and-drop, block-based machine learning tool that enables developers to apply machine learning techniques more efficiently. It transforms the traditional coding process into an intuitive visual interface, similar to building with Lego blocks.

Key Contributions:

- Lowered the technical barrier of ML development through visual abstraction of complex concepts
- Provided pre-built blocks for common machine learning tasks
- Simplified algorithm selection and configuration with guided model selection
- Implemented drag-and-drop data transformation to simplify data preprocessing
- Offered a streamlined workflow from data to deployment

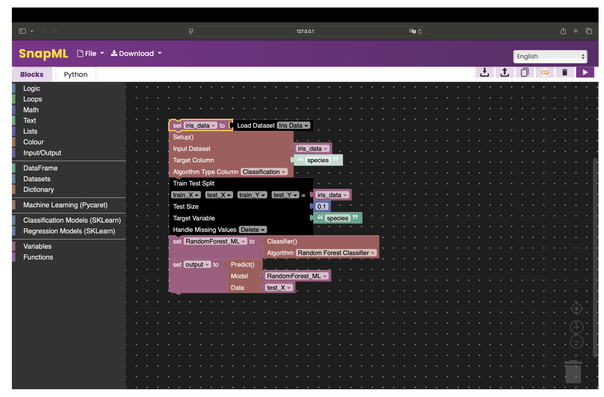
By providing a visual approach to machine learning development, SnapML addresses the steep learning curve typically associated with ML, making it accessible to a broader audience including:

- Software developers without ML expertise
- Data scientists seeking rapid prototyping
- Educators teaching ML concepts
- Cross-functional teams with varying technical backgrounds

2 Background & Motivation

With the rapid development of Machine Learning (ML), it has become an essential part of modern technology. However, many developers face significant barriers when attempting to implement ML solutions:

Challenges	SnapML Solutions
Complex mathematical theories	Visual abstraction of complex concepts
Extensive programming requirements	Pre-built blocks for common ML tasks
Algorithm selection difficulties	Guided model selection and configuration
Data preprocessing complexity	Drag-and-drop data transformation
Integration challenges	Streamlined workflow from data to deployment

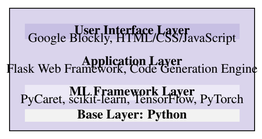


The need for a visual programming environment for ML arises from the growing demand for:

- Faster development cycles in data science projects
- Tools that facilitate learning and experimentation
- Accessible ML development for non-specialists
- Improved collaboration between technical and non-technical team members

3 Technology Stack

SnapML integrates several proven technologies to create a seamless machine learning development experience:

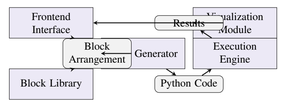


Key Technologies:

- **Google Blockly:** Open-source visual programming library that forms the foundation of our drag-and-drop interface
- **Flask:** Lightweight Python Web framework for serving the application
- **PyCaret:** Low-code ML library that powers many of the built-in models
- **Python:** The underlying programming language that executes the generated code

4 System Architecture

SnapML follows a modular architecture designed for extensibility and ease of use:



Components:

Frontend Interface: Provides the block workspace, preview panel, and settings controls

Block Library: Contains all available ML-specific blocks organized by category

Code Generator: Translates the visual block arrangement into executable Python code

Execution Engine: Runs the generated code on the server and handles data processing

Visualization Module: Renders results in various formats (charts, tables, metrics)

The data flow follows a clear path:

1. Users arrange blocks in the workspace
2. Block arrangements are translated to Python code
3. Code is executed on the server
4. Results are processed and visualized
5. Visualizations are displayed to the user

5 Case Study: Iris Classification

To demonstrate SnapML's capabilities, we implemented a classic ML task - the Iris flower classification:

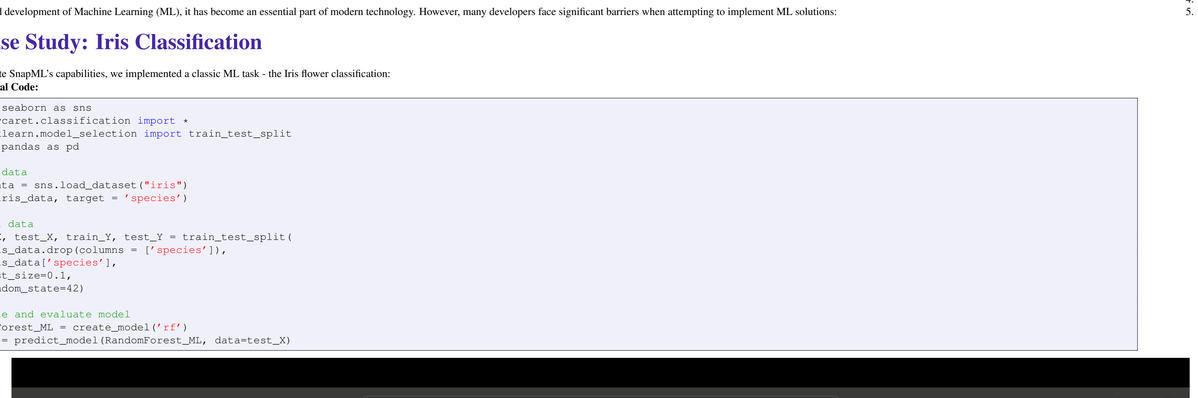
Traditional Code:

```python
import seaborn as sns
from pycaret.classification import *
from sklearn.model_selection import train_test_split
import pandas as pd

# Load data
iris_data = sns.load_dataset("iris")
setup(iris_data, target = 'species')

# Split data
train_X, test_X, train_Y, test_Y = train_test_split(
    iris_data.drop(columns = ['species']),
    iris_data['species'],
    test_size=0.1,
    random_state=42)

# Create and evaluate model
RandomForest_ML = create_model('rf')
output = predict_model(RandomForest_ML, data=test_X)
```



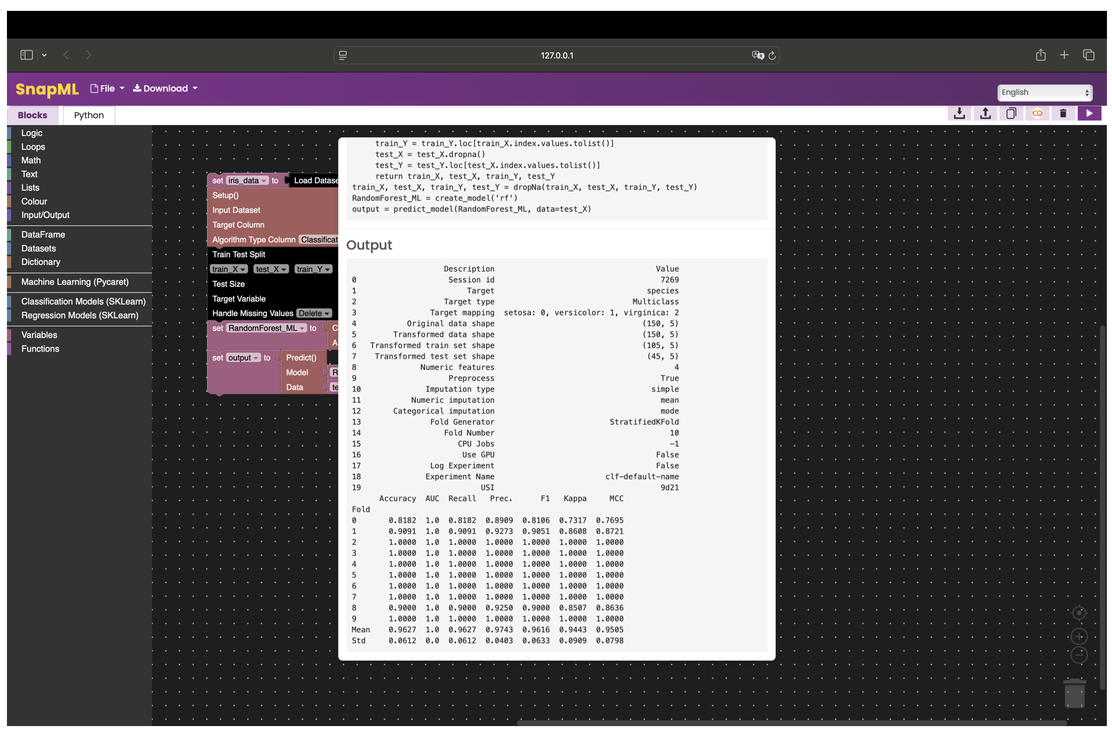

```python
import seaborn as sns
from pycaret.classification import *
from sklearn.model_selection import train_test_split
import pandas as pd
from sklearn.preprocessing import LabelEncoder, OneHotEncoder
import numpy as np

iris_data = None
train_X = None
test_X = None
train_Y = None
test_Y = None
RandomForest_ML = None
output = None


iris_data = sns.load_dataset("iris")
setup(iris_data, target = 'species')

train_X, test_X, train_Y, test_Y=train_test_split(iris_data.drop(columns = ['species']),iris_data['species'], test_size=0.1, random_state=42)
def dropNa(train_X, test_X, train_Y, test_Y):
    train_X = train_X.dropna()
    train_Y = train_Y.loc[train_X.index.values.tolist()]
    test_X = test_X.dropna()
    test_Y = test_Y.loc[test_X.index.values.tolist()]
    return train_X, test_X, train_Y, test_Y
train_X, test_X, train_Y, test_Y = dropNa(train_X, test_X, train_Y, test_Y)
RandomForest_ML = create_model('rf')
output = predict_model(RandomForest_ML, data=test_X)
```

With SnapML:



The case study demonstrates how SnapML simplifies the entire machine learning workflow:

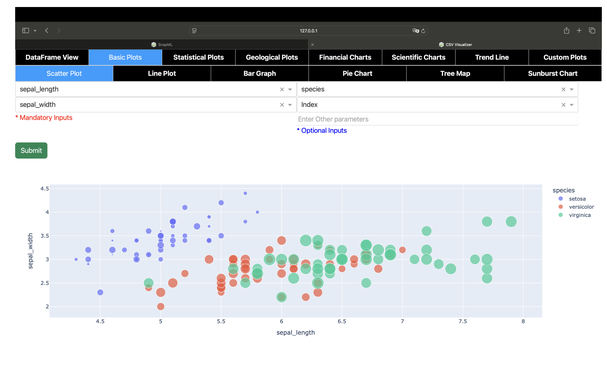
1. Data loading is accomplished with a single block
2. Data splitting and preprocessing require minimal configuration
3. Model selection is intuitive with visual parameter adjustment
4. Evaluation and visualization happen automatically

What would typically require dozens of lines of code and deep ML knowledge is reduced to a simple visual workflow that can be built in minutes.

6 Results & Benefits

SnapML delivers significant advantages for machine learning development:

Metric	Improvement (%)
Development Time	65%
Learning Curve	80%
Collaboration	70%
Iteration Speed	60%



Key Benefits:

Lower Learning Curve: Accessible to non-specialists and beginners

Visual Debugging: Easier to identify and fix issues in the workflow

Educational Value: Learn ML concepts through interactive visual experimentation

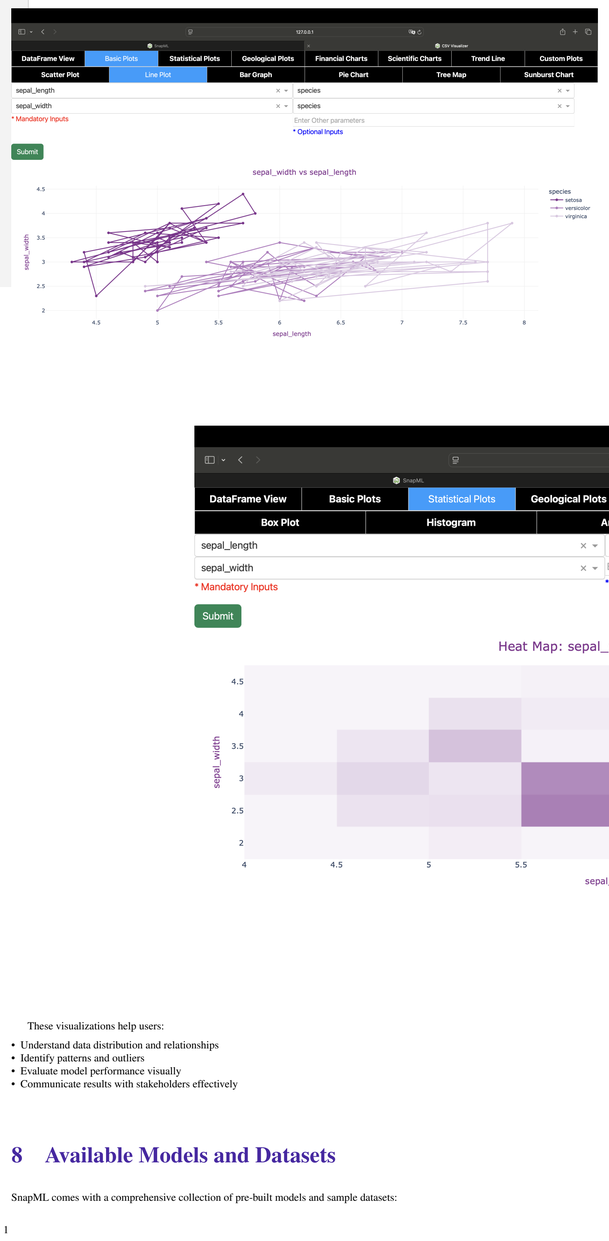
Team Collaboration: Bridge communication between technical and non-technical stakeholders

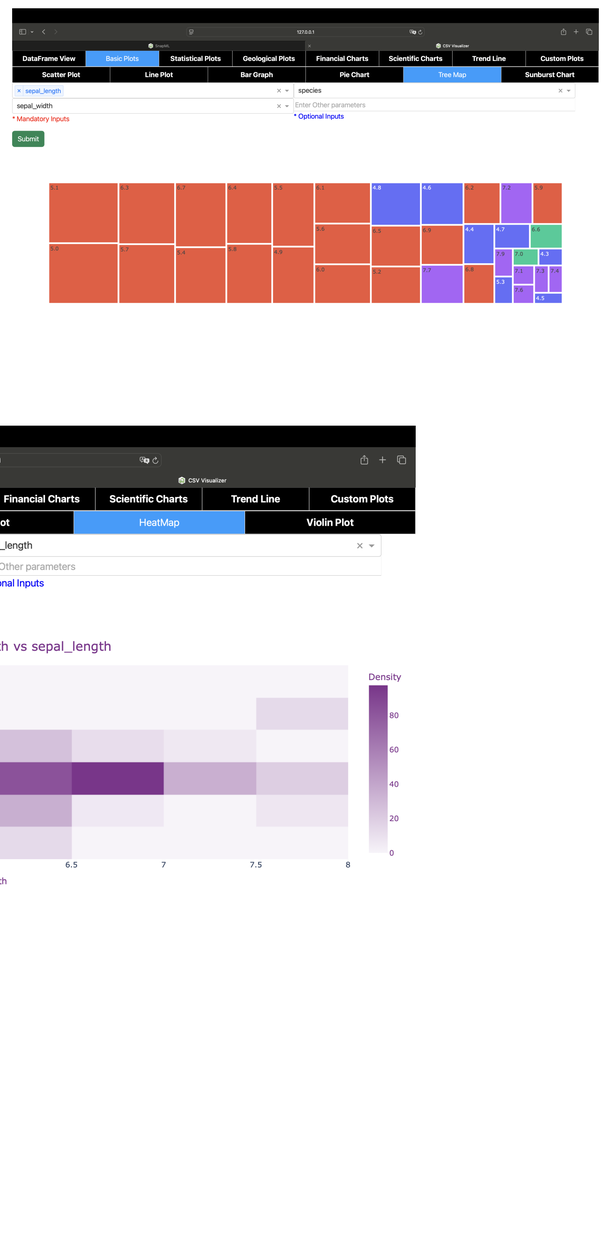
Rapid Prototyping: Build and test ML workflows in a fraction of the time

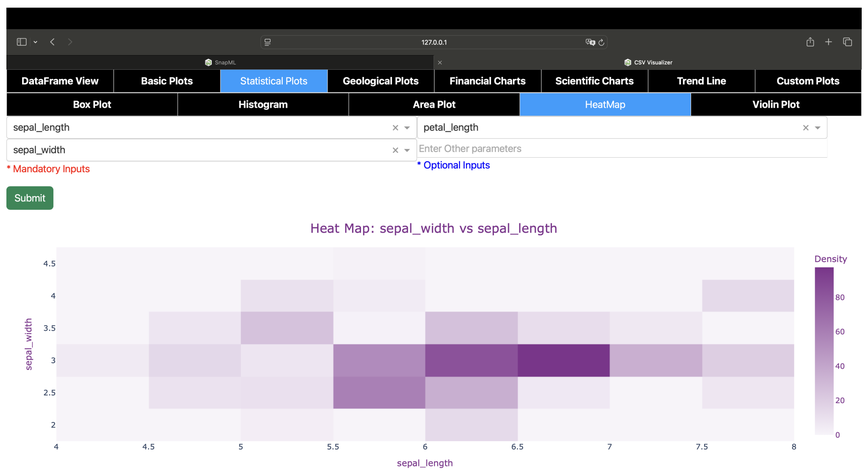
User feedback has consistently highlighted improved productivity and reduced barriers to entry for machine learning projects. Teams report being able to involve more stakeholders in the ML development process, leading to better alignment with business objectives.

7 Data Visualization Capabilities

SnapML provides powerful data visualization capabilities that help users understand their datasets and model performance:





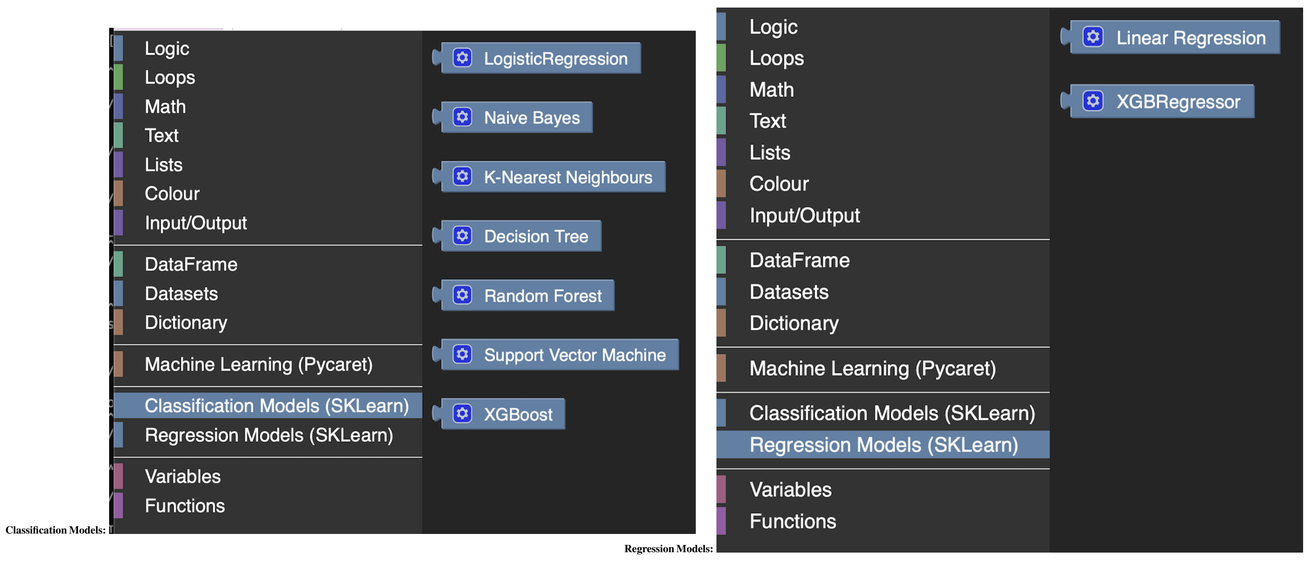


These visualizations help users:

- Understand data distribution and relationships
- Identify patterns and outliers
- Evaluate model performance visually
- Communicate results with stakeholders effectively

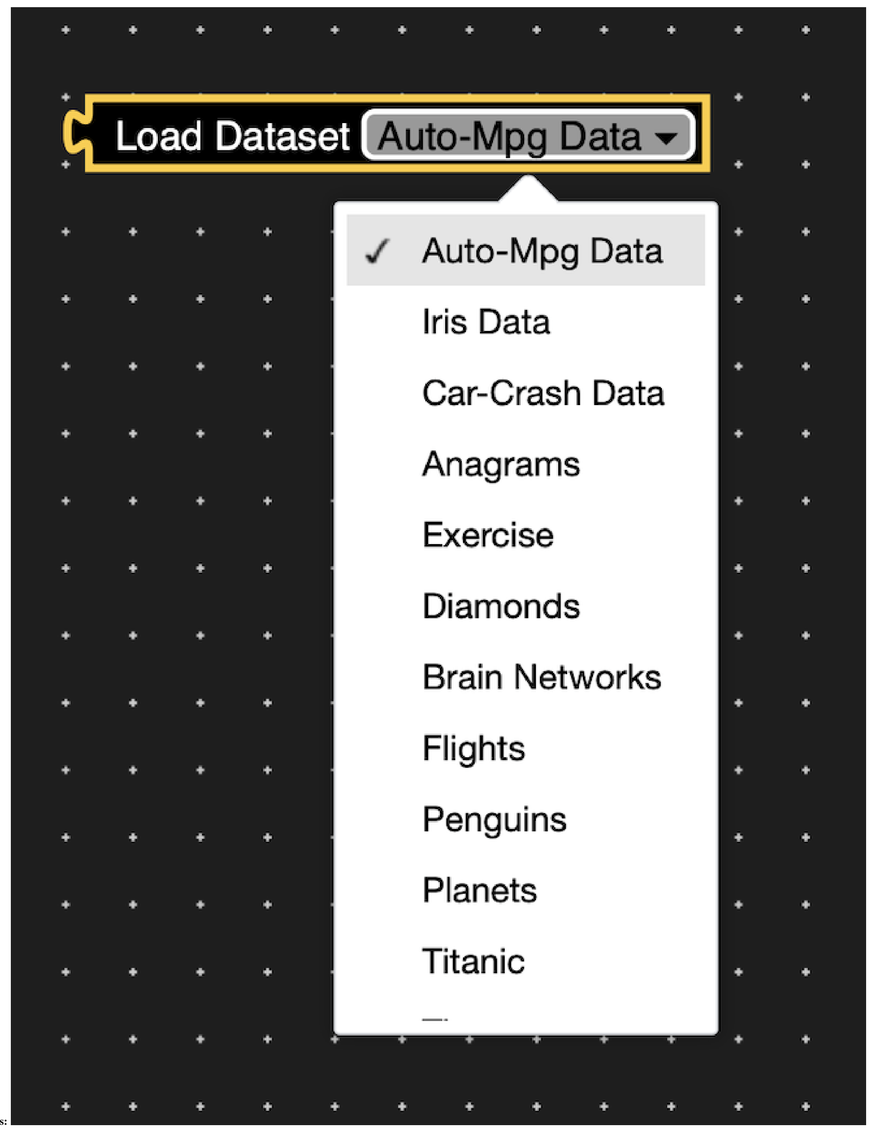
8 Available Models and Datasets

SnapML comes with a comprehensive collection of pre-built models and sample datasets:



Classification Models:

Regression Models:



Pre-loaded Datasets:

Having these resources readily available allows users to:

- Experiment with different algorithms quickly
- Learn from example datasets
- Compare model performance across different approaches
- Start prototyping without needing to source external data

9 Future Development & Conclusion

Future Development

Our roadmap for SnapML includes several exciting enhancements:

- **Enhanced Block Library:** Adding support for more advanced ML algorithms, including deep learning architectures and reinforcement learning
- **Cloud Integration:** Seamless deployment to cloud platforms such as AWS, Google Cloud, and Azure
- **Collaborative Features:** Real-time multi-user editing and version control integration
- **Custom Block Creation:** User-defined blocks for specialized tasks and domain-specific functionality
- **Mobile Support:** Responsive design for various devices to enable on-the-go ML development
- **Extended Visualization:** More interactive and customizable data visualization options

Conclusion

SnapML represents a significant step forward in democratizing machine learning development by:

- Making ML accessible to a broader audience regardless of technical background
- Reducing the complexity of building and testing ML models
- Providing an educational platform for learning ML concepts through practical application
- Enabling rapid prototyping and experimentation for data scientists
- Facilitating better collaboration between technical and business teams

Through its intuitive interface and powerful capabilities, SnapML aims to become an essential tool for developers looking to incorporate machine learning into their projects without the steep learning curve traditionally associated with ML development.