**ДНІПРОВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ОЛЕСЯ ГОНЧАРА**

Факультет прикладної математики

Кафедра обчислювальної математики та математичної кібернетики

ЗВІТ

про виконання лабораторної роботи №1

з дисципліни «Математичні основи інформаційних технологій»

Виконав:

студент групи ПМ-22м-1

Щербак Роман

Викладач:

О. С. Магас

Дніпро

2022

1. **Постановка задачі**

Створити експертну систему для передбачення погоди з використанням відповідного набору погодних даних. У роботі використати декілька різних методів передбачення.

1. **Опис розв’язку**

Для виконання поставленої задачі було використано набір даних із файлу weather.csv, що містить такі змінні: 'MinTemp', 'MaxTemp', 'Rainfall', 'Evaporation', 'Sunshine', 'WindGustDir', 'WindGustSpeed', 'WindDir9am', 'WindDir3pm', 'WindSpeed9am', 'WindSpeed3pm', 'Humidity9am', 'Humidity3pm', 'Pressure9am', 'Pressure3pm', 'Cloud9am', 'Cloud3pm', 'Temp9am', 'Temp3pm', 'RainToday', 'RISK\_MM', 'RainTomorrow'.

Було написано програмою мовою Python, що дозволяє користувачу за допомогою графічного інтерфейсу з бібліотеки tkinter обирати бажані змінні для роботи експертної системи, визначає ті незалежні змінні, з якими обрана залежна корелює найкраще (зі заданого інтервалу значень модуля кореляції, за замовчуванням установленого як (0.2;0.6) ), навчає модель регресії заданих порядків (за замовчуванням лінійну та квадратурну) та використовує її для передбачення значень неперервних змінних: 'MinTemp', 'MaxTemp', 'Rainfall', 'Evaporation', 'Sunshine', 'WindGustSpeed', 'WindDir9am', 'WindDir3pm', 'WindSpeed9am', 'WindSpeed3pm', 'Humidity9am', 'Humidity3pm', 'Pressure9am', 'Pressure3pm', 'Cloud9am', 'Cloud3pm', 'Temp9am', 'Temp3pm', 'RainToday', 'RISK\_MM', 'RainTomorrow' та модель дерева прийняття рішень для бінарної класифікації змінних RainToday та RainTomorrow і для мультикласової класифікації змінних WindDirection9am, WindDirection3pm та WindGustDir, з установленими за замовчуванням параметри глибини дерева: 5, мінімальної кількості значень із набору даних для поділу листка: 2, максимальної кількости листків у дереві: 10. Для роботи обох моделей набір даних ділиться на навчальну та тренувальну вибірку декількома різними способами, заданими змінною splits\_number, за замовчуванням установленою як 5.

Результати роботи видаються для кожного з різних поділів набору даних у форматі ‘Цикл {номер}: Час навчання: {час навчання}’. По завершенню роботи виводиться успішність роботи моделі у форматі: ‘Середня точність за {кількість поділів} поділів: {значення у відсотках} Середня похибка за {кількість поділів} поділів: число’, а дерево прийняття рішень додатково графічно будується в окремому файлі tree.png.

1. **Код програми**

* Модуль main.py

import prepare\_data as prd

import seaborn as sns

import graphic\_interface as gi

print()

# інтервал кореляції для включення незалежної змінної в модель, відкритий із обох боків

corr\_threshold = [0.2, 0.6]

# кількість різних поділів масиву даних

splits\_number = 5

# параметри для побудови дерева

tree\_parameters = {'max\_depth': 5, 'min\_samples\_leaf': 2, 'max\_leaf\_nodes': 10}

# максимальний степінь регресії

max\_regression\_pow = 2

dataset, continuous\_vars, discrete\_vars = prd.prepare\_dataset()

gi.main\_work(dataset, continuous\_vars, discrete\_vars, corr\_threshold, splits\_number, max\_regression\_pow, tree\_parameters)

* Модуль prepare\_data.py

import pandas as pd

import numpy as np

def prepare\_dataset():

dataset = pd.read\_csv('weather.csv').dropna()

variables = dataset.columns

continuous\_vars = [var for var in variables if dataset.dtypes[var] != "object"]

discrete\_vars = np.setdiff1d(variables, continuous\_vars)

for dv in dataset[discrete\_vars]:

if dataset[dv][0] in ['Yes', 'No']:

dataset[dv+"\_int"] = dataset[dv] == "Yes"

else:

wind\_set = set(dataset[dv])

wind\_dict\_reversed = dict(enumerate(wind\_set))

wind\_dict = {v:k for k,v in wind\_dict\_reversed.items()}

dataset[dv+"\_int"] = list(map(lambda x: wind\_dict.get(x), dataset[dv]))

return dataset, continuous\_vars, discrete\_vars

* Модуль graphic\_interface.py

import tkinter as tk

import sklearn.model\_selection as skms

import build\_model as bm

def main\_work(dataset, continuous\_vars, discrete\_vars, corr\_threshold, splits\_number, max\_regression\_pow, tree\_parameters):

root = tk.Tk()

root.title("Вибір змінних")

root.geometry('700x500')

choose\_continuous\_var(root, dataset, continuous\_vars, corr\_threshold, max\_regression\_pow, splits\_number)

choose\_discrete\_var(root, dataset, discrete\_vars, corr\_threshold, splits\_number, tree\_parameters)

end\_window(root)

root.mainloop()

def end\_window(root):

def button():

root.destroy()

submit\_button = tk.Button(root, text = 'Завершити роботу', command = lambda: button())

submit\_button.pack()

def get\_independent\_vars(dataset, var\_to\_predict, corr\_threshold, var\_type):

corr = dataset.corr()

if var\_type == "continuous":

var\_correlation = dict(corr.get(var\_to\_predict))

elif var\_type == "discrete":

var\_correlation = dict(corr.get(var\_to\_predict+"\_int"))

best\_var\_correlation = {k: v for k, v in var\_correlation.items() if abs(v) < corr\_threshold[1] and abs(v) > corr\_threshold[0]}

while len(best\_var\_correlation) == 0:

corr\_threshold[0] -= 0.1

corr\_threshold[1] += 0.1

best\_var\_correlation = {k: v for k, v in var\_correlation.items() if abs(v) < corr\_threshold[1] and abs(v) > corr\_threshold[0]}

return best\_var\_correlation

def prepare\_model\_parameters(dataset, var\_to\_predict, corr\_threshold, splits\_number, var\_type):

best\_var\_correlation = get\_independent\_vars(dataset, var\_to\_predict, corr\_threshold, var\_type)

X = dataset[best\_var\_correlation.keys()].values

if var\_type == "continuous":

Y = dataset[var\_to\_predict].values

elif var\_type == "discrete":

Y = dataset[var\_to\_predict+"\_int"].values

kf = skms.KFold(n\_splits = splits\_number, shuffle = True)

return best\_var\_correlation, X, Y, kf

def choose\_continuous\_var(root, dataset, continuous\_vars, corr\_threshold, max\_regression\_pow, splits\_number):

label = tk.Label(root, text="Оберіть неперервну змінну: ")

label.pack()

value\_inside = tk.StringVar()

cond\_entry = tk.OptionMenu(root, value\_inside, \*continuous\_vars)

cond\_entry.pack()

def get\_cont\_var():

cont\_var\_to\_predict = value\_inside.get()

print(f"Обрано змінну {cont\_var\_to\_predict}")

best\_cont\_var\_correlation, X, Y, kf = prepare\_model\_parameters(dataset, cont\_var\_to\_predict, corr\_threshold, splits\_number, "continuous")

print(f"Обрана змінна найкраще корелює зі змінними: {best\_cont\_var\_correlation}")

print()

regression\_powers = [i for i in range(1, max\_regression\_pow+1)]

regression\_names = {1: "Лінійна регресія", 2: "Квадратурна регресія", 3: "Кубічна регресія"}

for rp in regression\_powers:

print(regression\_names.get(rp, f"Регресія степеня {rp}"))

bm.build\_regression\_model(kf, X, Y, rp)

submit\_button = tk.Button(root, text='Обрати змінну', command = lambda: get\_cont\_var())

submit\_button.pack()

def choose\_discrete\_var(root, dataset, discrete\_vars, corr\_threshold, splits\_number, tree\_parameters):

label\_1 = tk.Label(root, text="Оберіть дискретну змінну: ")

label\_1.pack()

value\_inside\_1 = tk.StringVar()

cond\_entry\_1 = tk.OptionMenu(root, value\_inside\_1, \*discrete\_vars)

cond\_entry\_1.pack()

def get\_discr\_var():

discrete\_var\_to\_predict = value\_inside\_1.get()

print(f"Обрано змінну {discrete\_var\_to\_predict}")

best\_discrete\_var\_correlation, X, Y, kf = prepare\_model\_parameters(dataset, discrete\_var\_to\_predict, corr\_threshold, splits\_number, "discrete")

print(f"Обрана змінна найкраще корелює зі змінними: {best\_discrete\_var\_correlation}")

print()

print("Дерево прийняття рішень")

bm.build\_decision\_tree\_model(kf, X, Y, list(best\_discrete\_var\_correlation), tree\_parameters)

submit\_button = tk.Button(root, text='Обрати змінну', command = lambda: get\_discr\_var())

submit\_button.pack()

* Модуль build\_model.py

import sklearn.preprocessing as skp

import sklearn.linear\_model as sklm

import time as tm

import sklearn.metrics as skm

import numpy as np

import sklearn.tree as skt

import graphviz as gphv

import os

os.environ["PATH"] += os.pathsep + 'C:/Program Files/Graphviz/bin/'

def train\_evaluate\_model(model, X, Y, train, test, ind):

X\_train, Y\_train = X[train], Y[train]

X\_test, Y\_test = X[test], Y[test]

start = tm.perf\_counter()

model.fit(X\_train, Y\_train)

time = tm.perf\_counter()-start

print(f"Цикл {ind}:")

print(f"Час навчання {time}")

print()

prediction = model.predict(X\_test)

return X\_test, Y\_test, prediction

def build\_regression\_model(kf, X, Y, regression\_degree):

poly = skp.PolynomialFeatures(degree = regression\_degree, include\_bias = False)

X = poly.fit\_transform(X)

scores = []

errs = []

i = 0

for train, test in kf.split(X):

model = sklm.LinearRegression()

X\_test, Y\_test, prediction = train\_evaluate\_model(model, X, Y, train, test, i)

model\_performance = model.score(X\_test, Y\_test)

error = skm.mean\_squared\_error(Y\_test, prediction)

scores.append(model\_performance)

errs.append(error)

i += 1

print(f"Середня точність за {kf.n\_splits} поділів: {round(np.mean(scores), 3)\*100}%")

print(f"Середня похибка за {kf.n\_splits} поділів: {round(np.mean(errs), 3)}")

print()

def build\_decision\_tree\_model(kf, X, Y, features, tree\_parameters):

accuracies = []

precisions = []

recalls = []

i = 0

for train, test in kf.split(X):

model = skt.DecisionTreeClassifier(max\_depth=tree\_parameters.get('max\_depth', 5), min\_samples\_leaf=tree\_parameters.get('min\_samples\_leaf', 2), max\_leaf\_nodes=tree\_parameters.get('max\_leaf\_nodes', 10))

X\_test, Y\_test, prediction = train\_evaluate\_model(model, X, Y, train, test, i)

accuracy = skm.accuracy\_score(Y\_test, prediction)

try:

precision = skm.precision\_score(Y\_test, prediction)

recall = skm.recall\_score(Y\_test, prediction)

except ValueError:

precision = skm.precision\_score(Y\_test, prediction, average = 'micro')

recall = skm.recall\_score(Y\_test, prediction, average = 'micro')

accuracies.append(accuracy)

precisions.append(precision)

recalls.append(recall)

i += 1

print()

print(f"Точність (accuracy): {np.mean(accuracies)\*100}%")

print(f"Влучність (precision): {np.mean(precisions)\*100}%")

print(f"Відкликання/чутливість (recall/sensitivity): {np.mean(recalls)\*100}%")

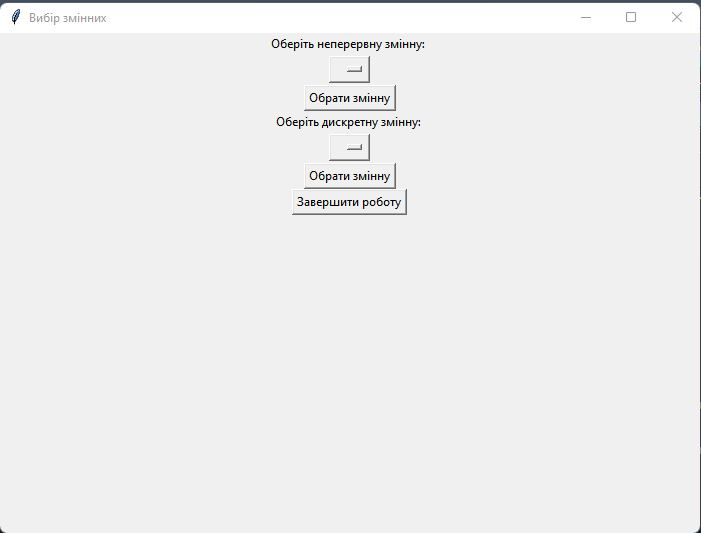
print()

dot\_file = skt.export\_graphviz(model, feature\_names=features)

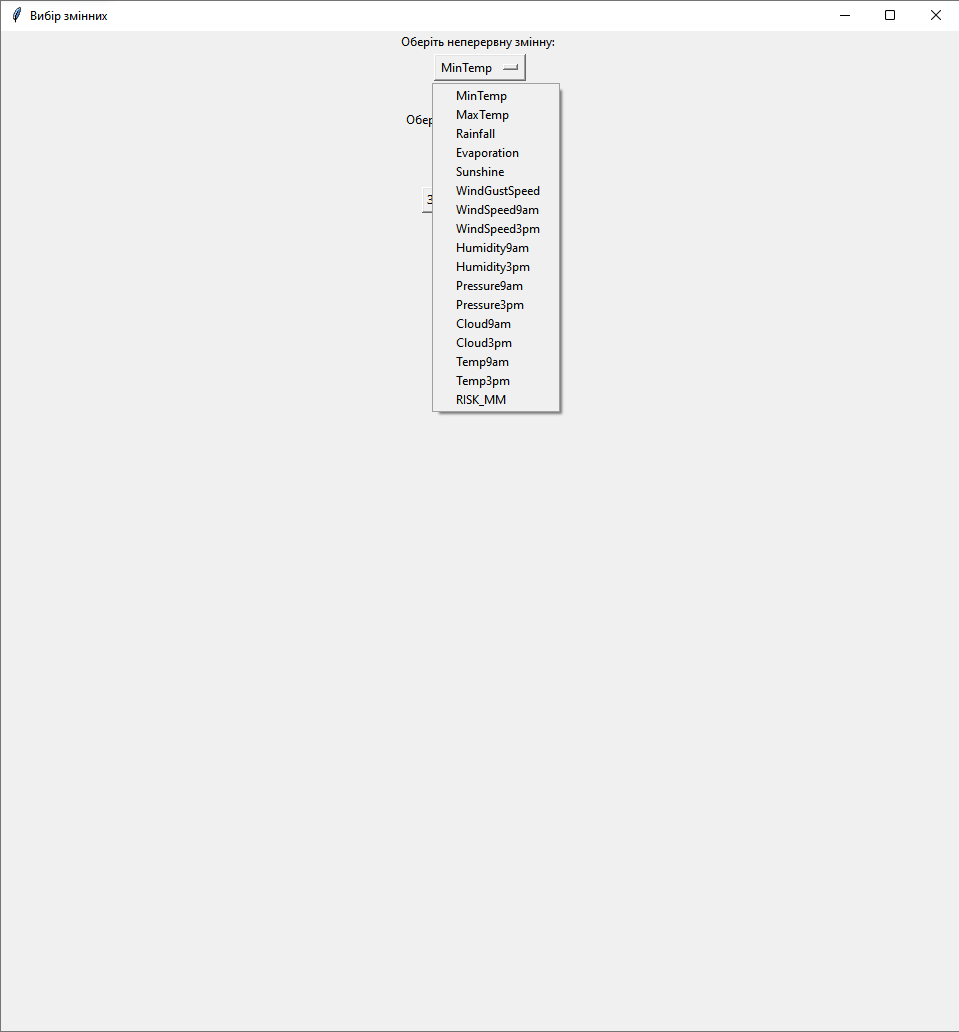
graph = gphv.Source(dot\_file)

graph.render(filename="tree", format = "png", cleanup=True)

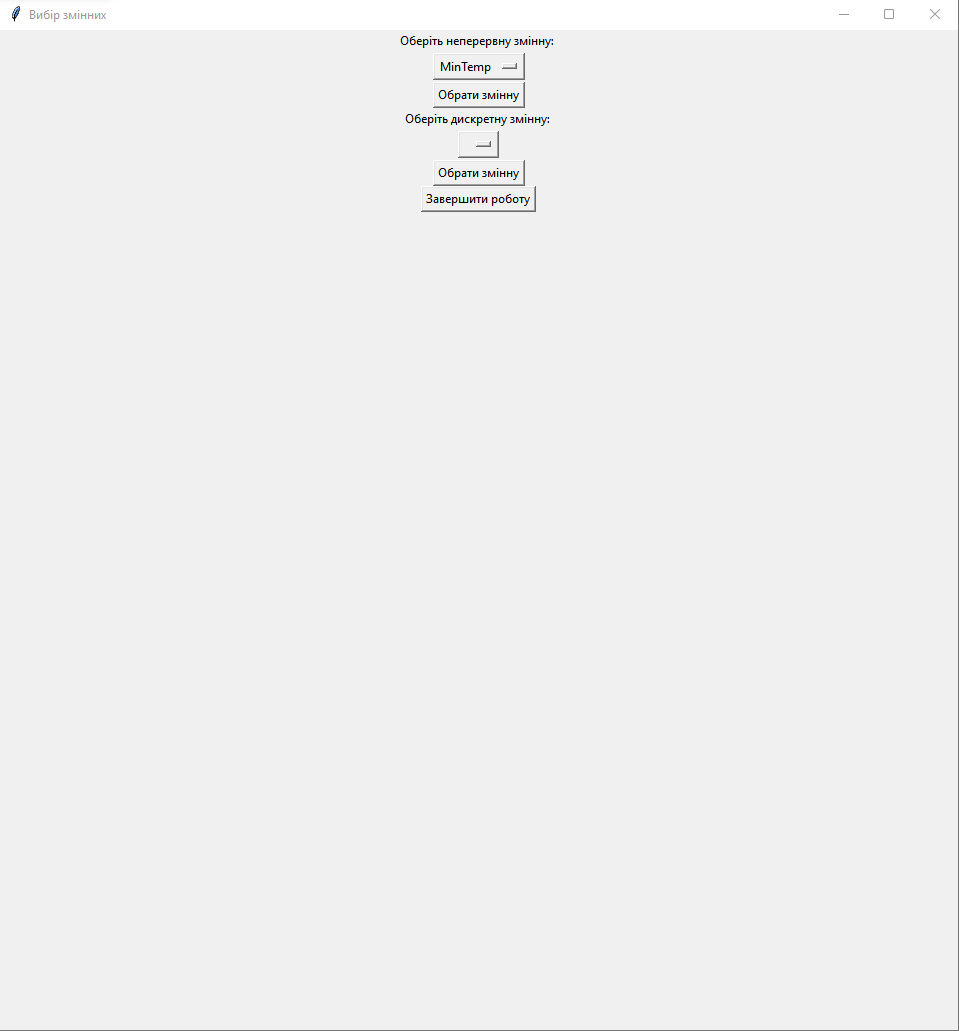
1. **Скриншоти роботи програми**



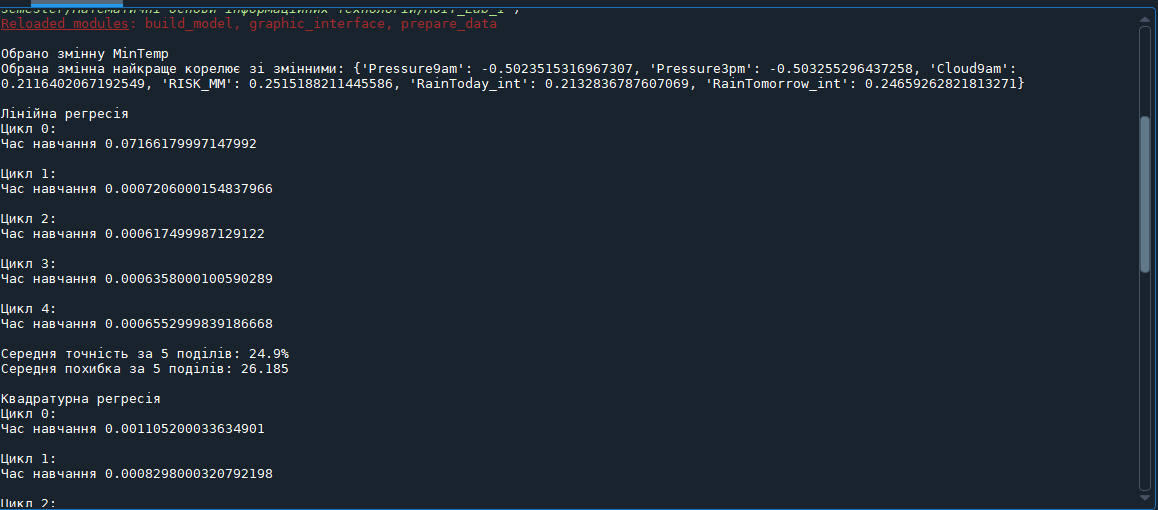
*Зображення 1.* Графічний інтерфейс користувача для вибору змінних



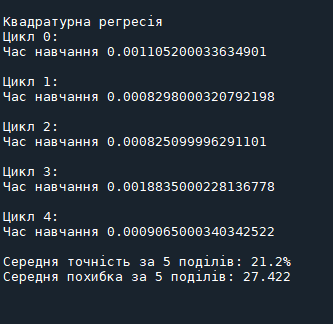
*Зображення 2.* Список неперервних змінних



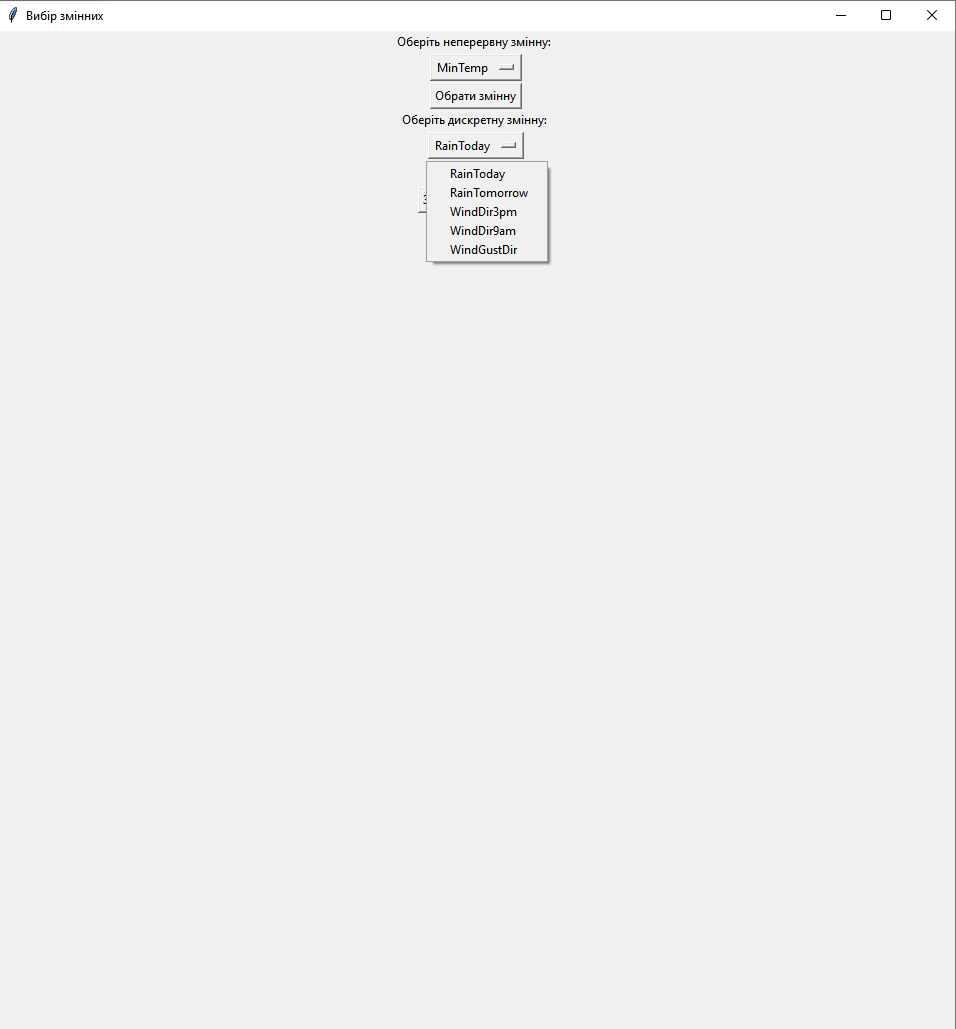
*Зображення 3.* Обрана неперервна змінна



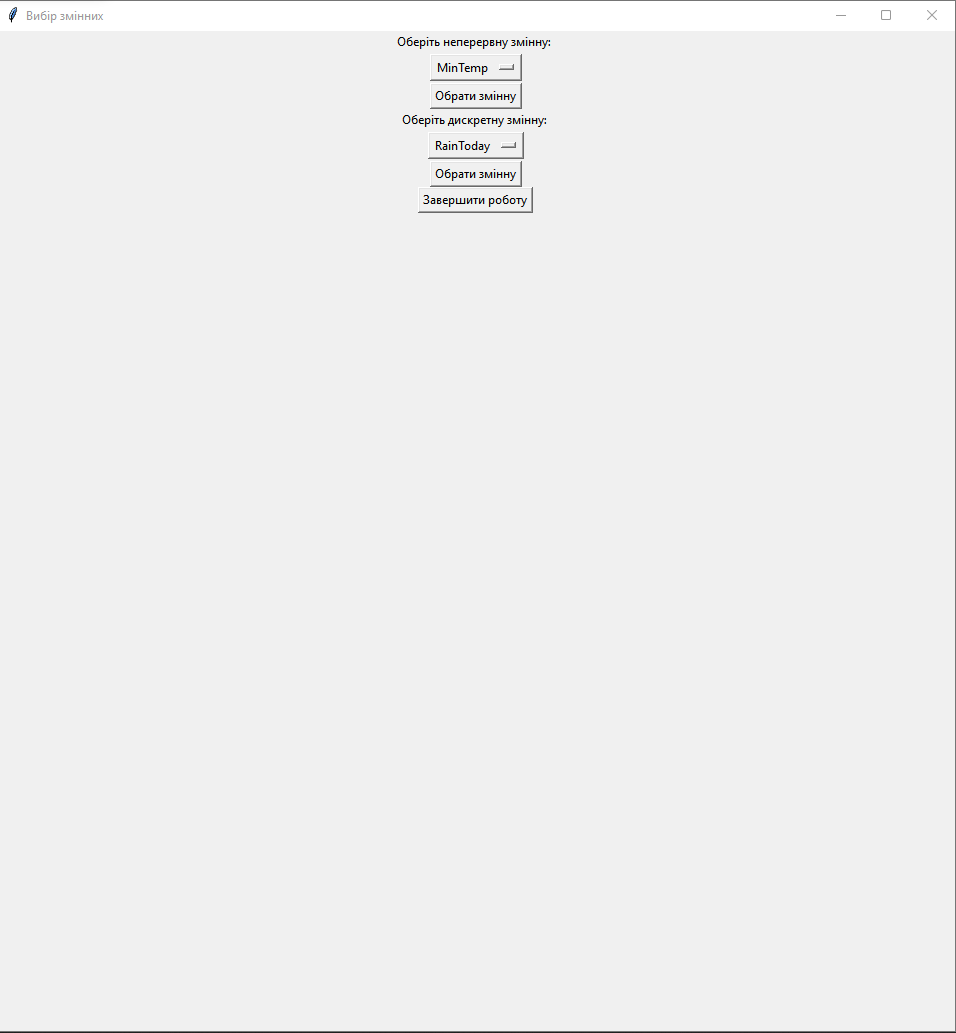
*Зображення 4.* Робота програми для обраної неперервної змінної



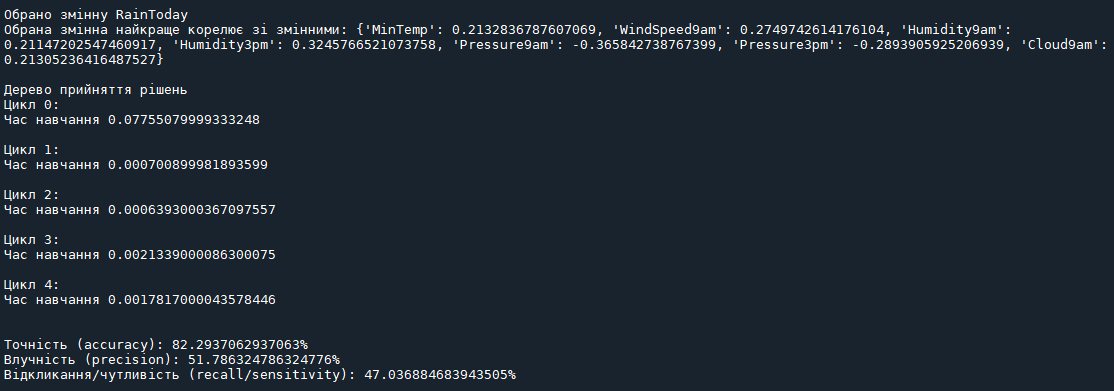
*Зображення 5.* Робота програми для обраної неперервної змінної



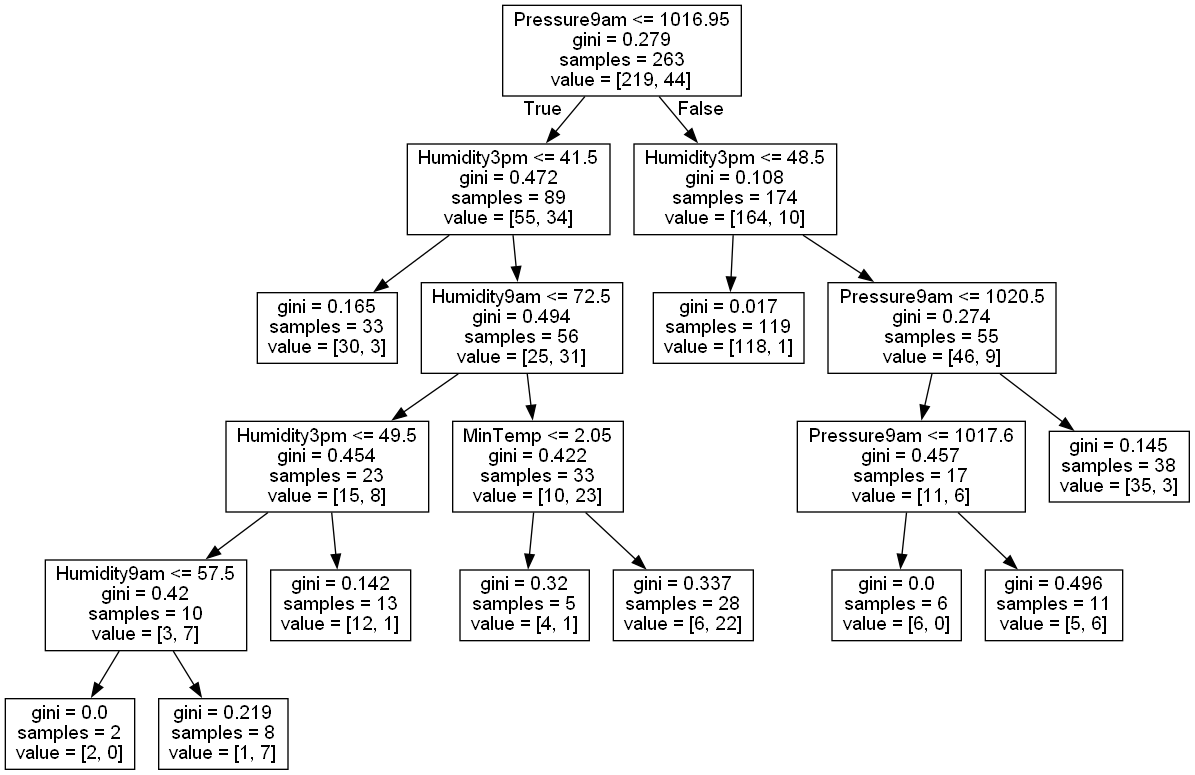
*Зображення 6.* Список дискретних змінних



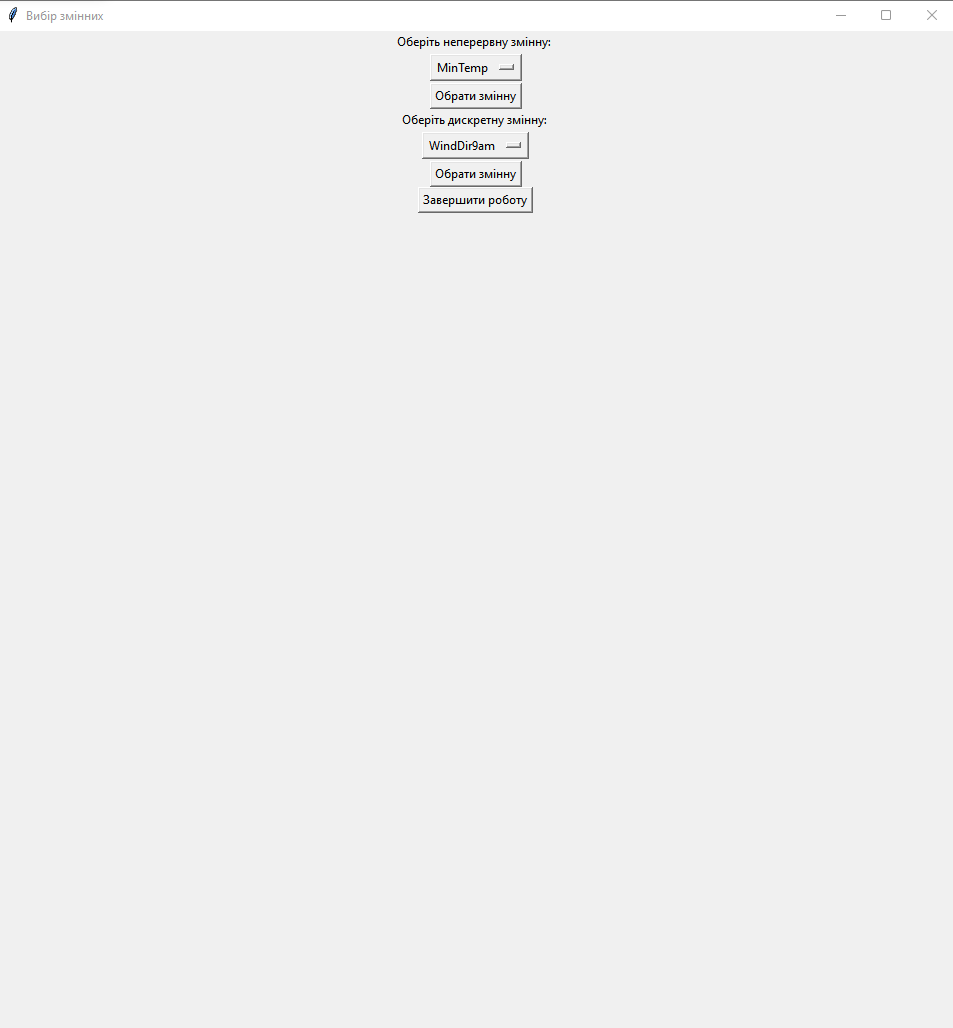
*Зображення 7.* Обрана дискретна бінарна змінна



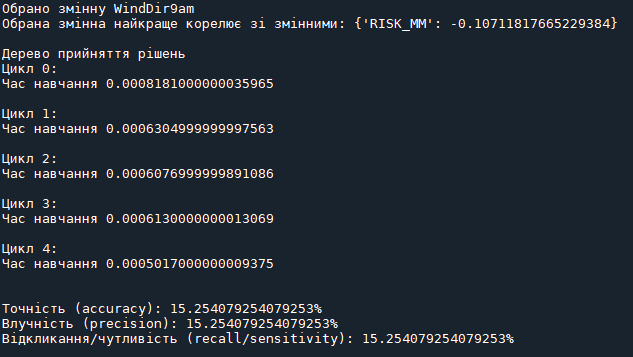
*Зображення 8.* Робота програми для обраної дискретної бінарної змінної



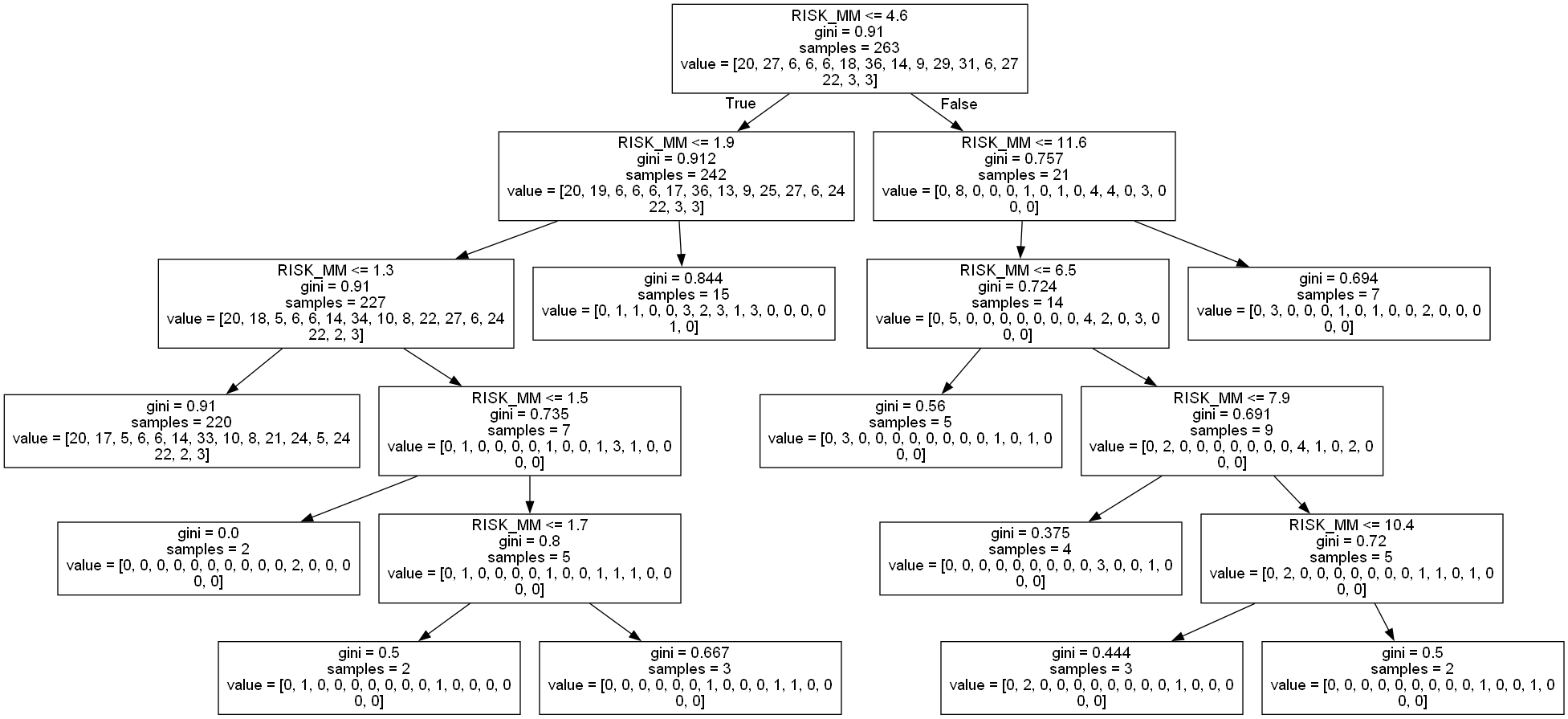
*Зображення 9.* Побудоване дерево прийняття рішень для обраної дискретної бінарної змінної



*Зображення 10.* Обрана дискретна мультикласна змінна



*Зображення 11.* Робота програми для обраної дискретної мультикласної змінної



*Зображення 12.* Побудоване дерево прийняття рішень для обраної дискретної мультикласної змінної

1. **Висновок**

У ході лабораторної роботи було побудовано дві моделі: регресію для неперервних змінних та дерево прийняття рішень – для дискретних, що дозволяють передбачати погоду за допомогою вибраного набору даних із файлу weather.csv.