Predicting the Walkability of Towns in America

By: Raghav Kamineni & Arnav Gupta

Project Statement

- Walkability Index (WI) is the accessibility of amenities by walking, as mostly urban regions should be more than just transportation-dominated pathways.
- WI is based on numerous broader factors
  - Housing Prices
  - Population and Land Density
  - Diversity of Land Utilization
  - Neighborhood Layouts and Designs
  - Location Accessibility
  - Transit Service
  - Employment
  - Individual Demographics
- Knowing the WI of a region is important

Description of dataset

- Dataset has 220,740 instances, 117 attributes; skewed to the right
- Numerous attributes had over 70% missing and/or default values
- Class attribute (WI) originally was quantitative continuous data from [1-20)
- We plan to discretize and bin the class attribute as shown in the next slide

Dataset

- Compiled by U.S. Environmental Protection Agency
- Derived from Smart Location Database (Geographical Factors)

Preprocessing (Python)

- Changed the class attribute within Excel to ‘NatWalkInd’
- Discretized and binned class attribute

Class Distribution

- 66,989 instances for AboveAvgWalkable (30.3%)
- 74,795 instances for BelowAvgWalkable (33.9%)
- 25,630 instances for MostWalkable (11.6%)
- 53,326 instances for LeastWalkable (24.1%)

```python
# PYTHON SCRIPT FOR EQUAL WIDTH BINNING AND DISCRETIZING CLASS ATTRIBUTE
import pandas as pd

df = pd.read_csv("ModifiedDataset.csv")

column = df.pop("Walkability_Index")
df["Walkability_Index"] = column

def categorize_walkability(value):
    if 1 <= value <= 5.75:
        return 'LeastWalkable'
    elif 5.75 < value <= 10.5:
        return 'BelowAvgWalkable'
    elif 10.5 < value <= 15.25:
        return 'AboveAvgWalkable'
    elif 15.25 < value <= 20:
        return 'MostWalkable'

df["Walkability_Index"] = df["Walkability_Index"].apply(categorize_walkability)
df.to_csv("ModifiedDataset2.csv", index=False)
```

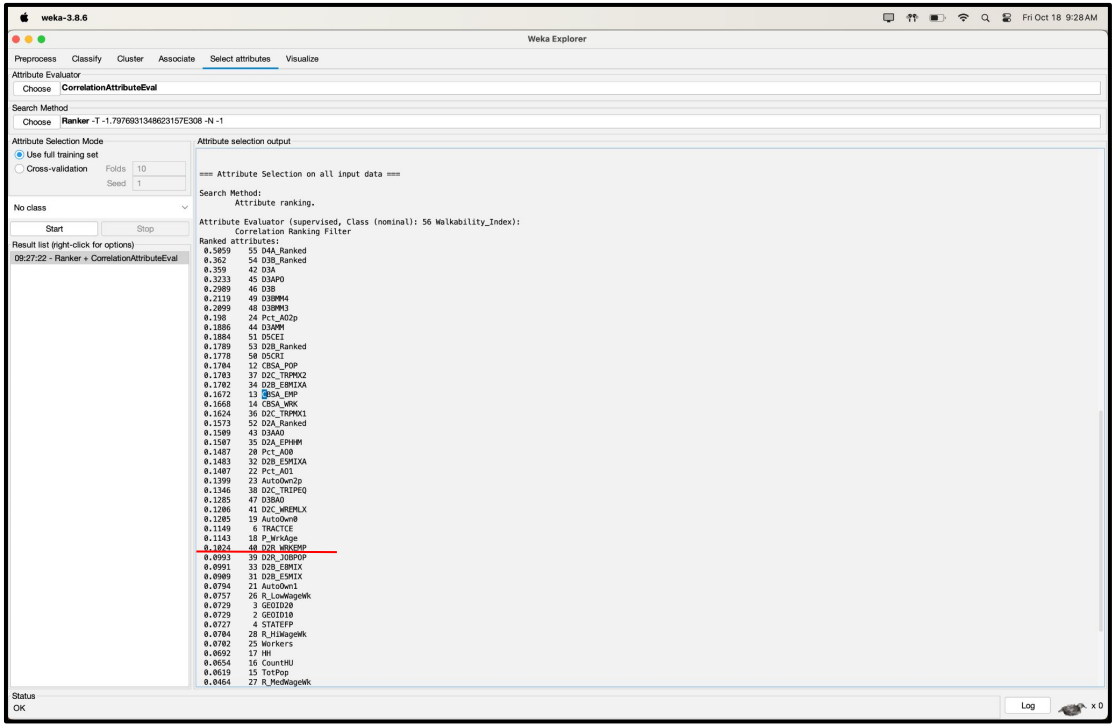
Preprocessing Cont'd (Python)

- Changed default values to missing with Python
- Removed attributes with more than 70% missing values
- Filled in remaining attributes missing values with mean/mode
- Left with 55 attributes

```
# PYTHON SCRIPT FOR REPLACING DEFAULT VALUES WITH MISSING VALUE REPRESENTATION
import pandas as pd

df = pd.read_csv("ModifiedDataset.csv")


for col in df.columns:
    df[col].replace(0, '?', inplace=True)
    df[col].replace(0.0, '?', inplace=True)
    df[col].replace('0', '?', inplace=True)
    df[col].replace('0.0', '?', inplace=True)

df.to_csv("ModifiedDataset2.csv", index=False)
```

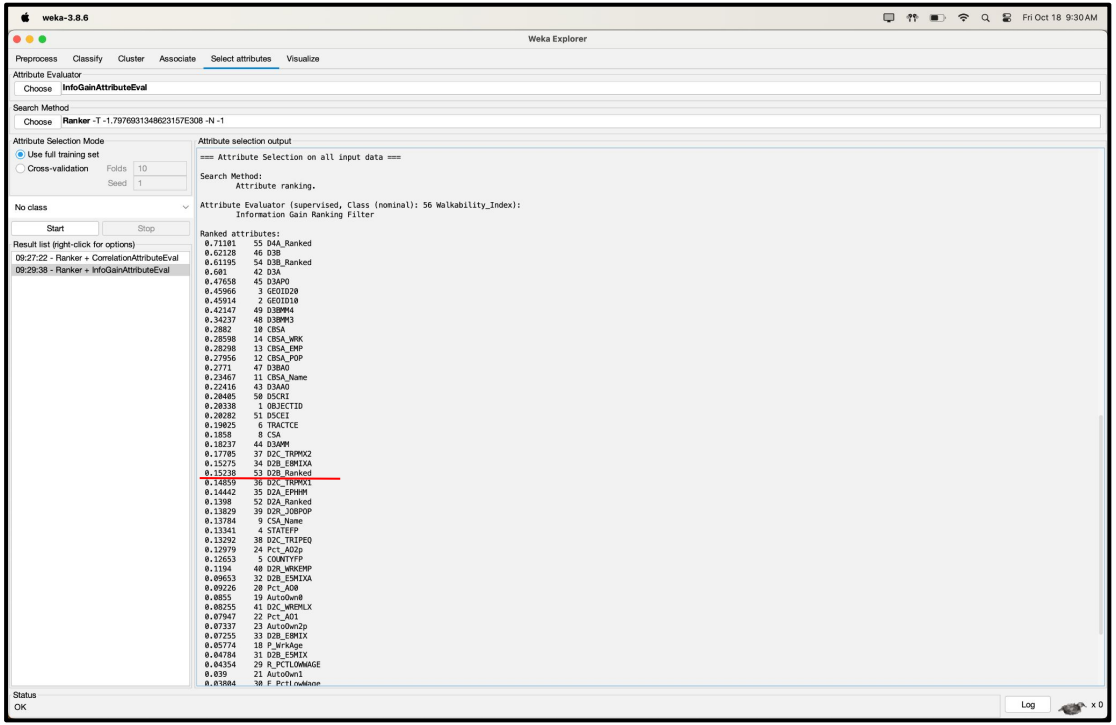
CorrelationAttributeEval

- Used Threshold of >= .1
- 32 instances left



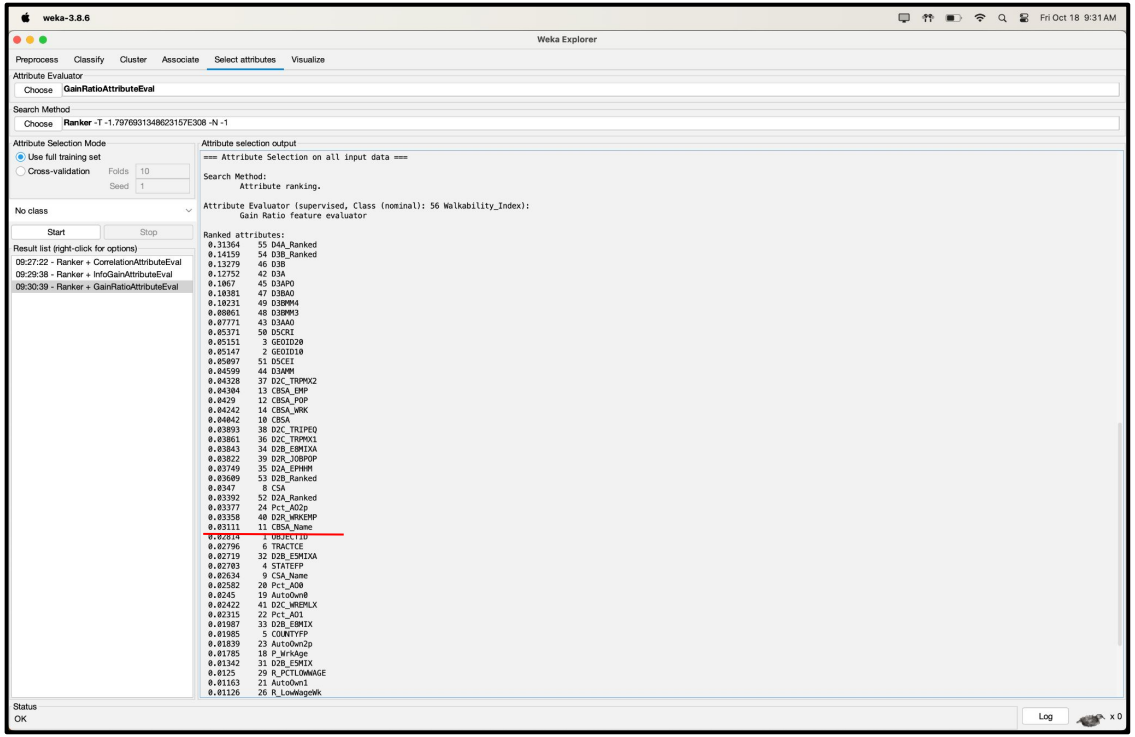
InfoGainAttributeEval

- Used Threshold of >= .15
- 25 instances left



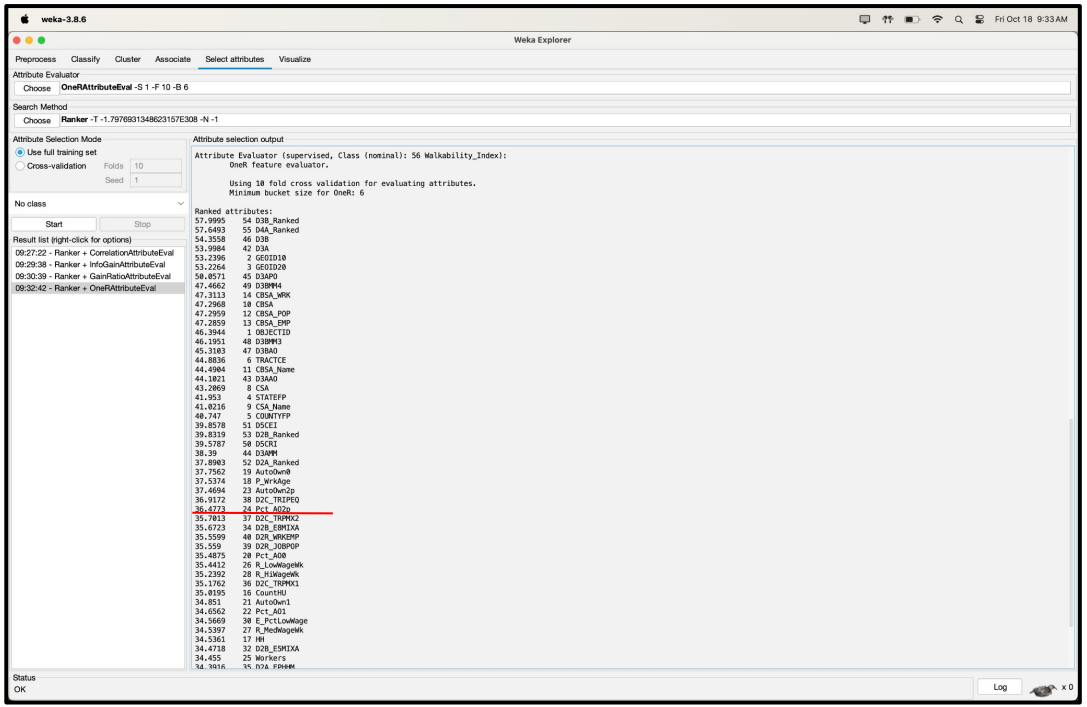
GainRatioAttributeEval

- Used Threshold of >= .03
- 30 instances left



OneRAttributeEval

- Used Threshold of >= 36
  - Min. 6 buckets
- 32 instances left



Independent Analysis

- Combined the best attributes based on visual metrics in WEKA, such as value distributions and easily visible trends through a combination of the best attributes of the last four attribute selection algorithms.
- Attribute list-> D4A_Ranked, D3B_Ranked, D3B, D3A, D3APO, D3BMM4, D3BMM3, Pct_AO2p, D5CEI, D2B_Ranked, CBSA_POP, D2C_TRPMX2, D2B_E8MIXA, CBSA_EMP, CBSA_WRK, D3BAO, D3AAO, D5CRI, D3AMM, D2C_TRIPEQ, D2R_WRKEMP, TRACTCE, CBSA_Name, CSA, AutoOwn0, and P_WrkAge.
- 26 instances left

Classifier Models

- Naive Bayes (Bayes)
  - Probabilistic classifier model based on Bayes Theorem (Conditional Probabilities) and feature-independent accuracy tests
- OneR (Rules)
  - Simpler probabilistic rule-based classifier model that uses a single rule from only the most informative attribute
- J48 (Trees)
  - Decision tree algorithm that builds tree based on splitting data by attribute values to classify instances through comprehensive training
- RandomForest (Trees)
  - Builds multiple decision trees during long training period and effectively captures general trends to prevent model overfitting

Training/Validation/Test Sets

- Utilized Python script to implement the Stratified Random Sampling technique due to uneven class distributions
- Split Data into Training/Validation/Testing
  - Training: 70%
  - Validation: 15%
  - Testing: 15%
- Created 15 files (five attribute selection algorithms and one training, validation, and testing dataset for each)
- Class Distribution preserved

```python
# PYTHON SCRIPT FOR STRATIFIED RANDOM SAMPLING SYSTEM
import pandas as pd
from sklearn.model_selection import train_test_split
import os


def stratified_sample_and_save(file_path,name):
    df = pd.read_csv(file_path)

    X = df.iloc[:, :-1]
    y = df.iloc[:, -1]

    X_train_temp, X_temp, y_train_temp, y_temp = train_test_split(
        X, y, test_size=0.3, stratify=y, random_state=42)

    X_val, X_test, y_val, y_test = train_test_split(
        X_temp, y_temp, test_size=0.5, stratify=y_temp, random_state=42)

    train_df = pd.concat([X_train_temp, y_train_temp], axis=1)
    val_df = pd.concat([X_val, y_val], axis=1)
    test_df = pd.concat([X_test, y_test], axis=1)


    train_df.to_csv(f"{name}_train.csv", index=False)
    val_df.to_csv(f"{name}_val.csv", index=False)
    test_df.to_csv(f"{name}_test.csv", index=False)


csv_files = ['ModifiedDatasetCORR.csv', 'ModifiedDatasetGAINRATIO.csv',
'ModifiedDatasetINFOGAIN.csv', 'ModifiedDatasetONER.csv',
'ModifiedDatasetINDIVIDUAL.csv']
names = ["corr","gainratio","infogain","oneR","indi"]

for file_path,x in zip(csv_files, range(5)):
    stratified_sample_and_save(file_path,names[x])
```

Results (All 20 Models)

Q1 Project Results: Classifier Models' Accuracy (20)		Classification Models (4)			
		Naive Bayes	OneR	J48	RandomForest
Attribute Selection Algorithms (5)	Correlation	0.7098	0.5424	0.9457	0.9510
	InfoGain	0.6956	0.5309	0.9061	0.8985
	GainRatio	0.7682	0.5315	0.9552	0.9466
	OneR	0.6986	0.5294	0.9579	0.9228
	Team Choice	0.6876	0.5300	0.9175	0.9146

Results (Best Model)

```
=== Summary ===

Correctly Classified Instances       31716               95.7869 %
Incorrectly Classified Instances      1395                4.2131 %
Kappa statistic                          0.9409
Mean absolute error                      0.0251
Root mean squared error                  0.1382
Relative absolute error                  7.0462 %
Root relative squared error             32.7121 %
Total Number of Instances            33111

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.942    0.020    0.936      0.942    0.939      0.920    0.978     0.931     AboveAvgWalkable
                 0.962    0.019    0.963      0.962    0.962      0.943    0.983     0.964     BelowAvgWalkable
                 0.911    0.008    0.924      0.911    0.917      0.908    0.976     0.893     MostWalkable
                 0.980    0.010    0.979      0.980    0.979      0.970    0.992     0.980     LeastWalkable
Weighted Avg.    0.958    0.015    0.958      0.958    0.958      0.943    0.984     0.954

=== Confusion Matrix ===

    a     b     c     d   <-- classified as
 7388   205   248     0 |     a = AboveAvgWalkable
  210 10906     0   225 |     b = BelowAvgWalkable
  292     0  2995     0 |     c = MostWalkable
    0   215     0 10427 |     d = LeastWalkable
```

OneR AttributeEval + J48 Classifier Model

Discussion and Conclusion

Successes

- Preprocessed data through Python and WEKA
- Conducted attribute selection algorithms in WEKA
- Tested classifier models with high accuracy

Best Model: OneR AttributeEval + J48 Classifier (95.79% accuracy)

Future Improvements

- Conduct PCA to reduce attributes
- Increase thresholds for attribute selection algorithms
- Optimize validation data effectiveness towards models
- Research best classifier models for our project type

Sources and Acknowledgements

Dataset: https://catalog.data.gov/dataset/walkability-index7 (Walkability Index dataset courtesy of U.S. Environmental Protection Agency)

Softwares: VSCode (Python) and WEKA

We'd like to thank the U.S. Environmental Protection Agency for compiling the dataset, and especially John Thomas for allowing public use of the dataset.

Thank you.