CPS 844 Research Paper Assignment

CPS 844

Winter 2025

Professor: Cherie Ding

Date: April 5, 2025

Mithuusan Kirupananthan (501102123)

Parvir Jhooty (501087853)

In this report, we will explore the Car Evaluation Dataset obtained from the UCI Machine Learning Repository. This dataset is composed of 1,728 instances and 6 categorical attributes. The target variable with this dataset represents the acceptability value of the car, being either unacceptable, acceptable, good, or very good (unacc, acc, good, vgood). The attributes used to evaluate the car's capability include:

1. Buying Price: (vhigh, high, med, low)
2. Maintenance Price: (vhigh, high, med, low)
3. Number of Doors: (2, 3, 4, 5)
4. Person Capacity: (2, 3, 4, 5)
5. Luggage trunk size: (small, med, big)
6. Safety rating: (low, med, high)

The problem addressed with this dataset is a classification problem, where the goal is to accurately classify the acceptability of a car based on the given attributes. Classification problems are seen mainly in areas such as medical diagnosis, fraud detection, and more. The main goal of this report is to analyze different algorithm approaches to determine which one performs the best at predicting the target variables based on the given attributes. With this dataset's target and given attributes, we will be determining which algorithm provides the most accurate acceptability class of the given vehicle attributes.

The first step to solving this classification problem is performing an explorative analysis on the dataset to understand it and identify any inconsistencies/missing values. We also need to preprocess the data if necessary to help with our analysis.

First, we read the dataset file into our Jupyter notebook, took note of the data type information of each attribute, and verified the number of rows and columns as seen below. There was also a check to ensure that there were no missing values in the dataset, ensuring that our algorithm tests later on will be accurate.

```
 #   Column   Non-Null Count  Dtype
---  ------   --------------  -----
 0   vhigh    1726 non-null   object
 1   vhigh.1  1726 non-null   object
 2   2        1726 non-null   int64
 3   2.1      1726 non-null   int64
 4   small    1726 non-null   object
 5   med      1726 non-null   object
 6   unacc    1726 non-null   object
dtypes: int64(2), object(5)
memory usage: 94.5+ KB
```

The dataset utilized the object data type for all the string attribute values and int64 for the integer value attributes. After discovering the data types, we saw that there were no column names for the data values. We applied column names to all the attributes and output the first few columns.

```
     buy price maintain price  doors  persons luggage_storage safety  class
0        vhigh          vhigh      2        2           small   high  unacc
1        vhigh          vhigh      2        2             med    low  unacc
2        vhigh          vhigh      2        2             med    med  unacc
3        vhigh          vhigh      2        2             med   high  unacc
4        vhigh          vhigh      2        2             big    low  unacc
...        ...            ...    ...      ...             ...    ...    ...
1721       low            low      5        5             med    med   good
1722       low            low      5        5             med   high  vgood
1723       low            low      5        5             big    low  unacc
1724       low            low      5        5             big    med   good
1725       low            low      5        5             big   high  vgood

[1726 rows x 7 columns]
```

Once the explorative analysis was completed, we pre-processed the data by changing the string attribute values to numerical values. This step is vital as the numeric values are needed for the algorithm analysis we will be conducting.

	buy price	maintain price	doors	persons	luggage_storage	safety	class
0	3	3	2	2	0	2	0
1	3	3	2	2	1	0	0
2	3	3	2	2	1	1	0
3	3	3	2	2	1	2	0
4	3	3	2	2	2	0	0

This was the dataset after applying the conversion, i.e. (low=0, med=1, high=2, vhigh=3). With these new numerical assignments to the attribute values, we can visualize and perform analysis on our data properly.

The first algorithm we used was a decision tree model. We chose this algorithm as it is easy to visualize and interpret the given data. In our case, with the car dataset, this algorithm will give us a clear map of the target variable assignment process that is easy to follow. The first compilation of the algorithm was done without feature selection, and a second compilation was done using feature selection.

Without Feature Selection:

```
Decision Tree Without Feature Selection
Accuracy Score: 0.8786127167630058

Classification Report:
               precision    recall  f1-score   support

           0       0.97      0.94      0.95       250
           1       0.71      0.86      0.78        71
           2       0.00      0.00      0.00        14
           3       0.50      0.82      0.62        11

    accuracy                           0.88       346
   macro avg       0.54      0.65      0.59       346
weighted avg       0.86      0.88      0.87       346
```

With Feature Selection:

```
Feature Importances:

           Feature  Importance
5           safety    0.315559
1   maintain price    0.203368
3          persons    0.184901
4  luggage_storage    0.133602
0        buy price    0.103241
2            doors    0.059328

Decision Tree With Feature Selection
Accuracy Score: 0.8786127167630058

Classification Report:
               precision    recall  f1-score   support

           0       0.97      0.94      0.95       250
           1       0.71      0.86      0.78        71
           2       0.00      0.00      0.00        14
           3       0.50      0.82      0.62        11

    accuracy                           0.88       346
   macro avg       0.54      0.65      0.59       346
weighted avg       0.86      0.88      0.87       346
```

Both of the decision tree models were trained and evaluated to classify the car evaluation classes into either unacc, acc, good, or vgood. As seen in the photos above both models achieve an accuracy score of approximately 88%. They both also had the same precision, recall, and f1-score of 0.86, 0.88, and 0.87. These high scores indicate that the decision tree model is performing well, and the model is well-balanced between precision and recall. The feature selection model also indicated that the feature with the highest importance is safety. However, the less important features, such as buy price and doors, did not negatively impact the accuracy of the tree model. Therefore, in the case of a decision tree model, the differentiation between a model with and without feature selection is non-existent, showing that both models have the same performance.

The next Algorithm we used was the random forest model. This algorithm handles the categorical data we provided without errors, and it prevents overfitting by creating multiple

decision trees. When we ran the algorithm with and without feature selection, we got the following results:

Without Feature Selection:

```
Random Forest Without Feature Selection
Accuracy Score: 0.9770114942528736

Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        64
           1       0.94      1.00      0.97        16
           2       1.00      0.75      0.86         4
           3       0.67      0.67      0.67         3

    accuracy                           0.98        87
   macro avg       0.90      0.85      0.87        87
weighted avg       0.98      0.98      0.98        87
```

With Feature Selection:

```
Feature Importances:

          Feature  Importance
5          safety    0.315559
1   maintain price    0.203368
3         persons    0.184901
4  luggage_storage    0.133602
0       buy price    0.103241
2           doors    0.059328

Random Forest With Feature Selection
Accuracy Score: 0.9770114942528736

Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        64
           1       0.94      1.00      0.97        16
           2       1.00      0.75      0.86         4
           3       0.67      0.67      0.67         3

    accuracy                           0.98        87
   macro avg       0.90      0.85      0.87        87
weighted avg       0.98      0.98      0.98        87
```

As seen in the photos above, both methods provide the same accuracy score of 97%, indicating that feature selection is unnecessary as the given attributes all play an important role in determining the target variable. In comparison to the decision tree model, it can also be seen that the random forest algorithm has a higher precision, recall, and f1-score of 0.98 compared to the decision trees' 0.88. This proves that the random forest algorithm is more accurate than the

decision tree algorithm and will provide better results. All in all, the Random forest algorithm provides much more accurate results when compared to the decision tree model.

Another algorithm that we tested was the K-Nearest Neighbor algorithm. We chose this algorithm as it works well with smaller datasets like the one we are analyzing. This algorithm is also parametric, while Random Forest is non-parametric, giving us a great comparison between both types. The results of the algorithm can be seen below:

Without Feature Selection:

```
K-Nearest Neighbors Classifier
Accuracy Score: 0.9508670520231214

Classification Report:
              precision    recall  f1-score   support

           0       0.99      0.98      0.98       250
           1       0.85      0.96      0.90        71
           2       0.75      0.64      0.69        14
           3       1.00      0.73      0.84        11

    accuracy                           0.95       346
   macro avg       0.90      0.83      0.85       346
weighted avg       0.95      0.95      0.95       346
```

With Feature Selection:

```
Feature Importances:

           Feature  Importance
5           safety    0.302062
3          persons    0.244511
0        buy price    0.153135
1   maintain price    0.145160
4  luggage_storage    0.095494
2            doors    0.059639

K-Nearest Neighbors Classifier (With Feature Selection)
Accuracy Score: 0.9508670520231214

Classification Report:
              precision    recall  f1-score   support

           0       0.99      0.98      0.98       250
           1       0.85      0.96      0.90        71
           2       0.75      0.64      0.69        14
           3       1.00      0.73      0.84        11

    accuracy                           0.95       346
   macro avg       0.90      0.83      0.85       346
weighted avg       0.95      0.95      0.95       346
```

After analyzing the results of running the KNN utilizing both methods, our final result is an accuracy of 95% from both. This indicates again that feature selection is unnecessary as all the given attributes are important to obtaining a target variable. The precision, recall, and f1-score are all 0.95, indicating that the algorithm has a high success rate of prediction on the dataset. However, when comparing back to the Random Forest Algorithm, it does not perform as well, telling us that Random Forest is the more accurate algorithm to use for predicting the target variable. In conclusion, although the KNN algorithm provides very accurate predictions, it still does not perform better than the Random Forest algorithm.

Naive Bayes is a probabilistic classifier that applies Bayes' Theorem with strong independence assumptions between features. It is well-suited for categorical data and is known for its simplicity and efficiency. In this project, we used the Categorical Naive Bayes model on the full dataset. Again, we used a reduced feature set selected from feature importance scores from a Random Forest classifier.

Naive Bayes Without Feature Selection
Accuracy Score: 0.7630057803468208

Classification Report:

	precision	recall	f1-score	support
0	0.92	0.92	0.92	250
1	0.56	0.31	0.40	71
2	0.29	0.14	0.19	14
3	0.22	1.00	0.36	11
accuracy			0.76	346
macro avg	0.50	0.59	0.47	346
weighted avg	0.80	0.76	0.76	346

Feature Importances:

	Feature	Importance
5	Feature 5	0.302062
3	Feature 3	0.244511
0	Feature 0	0.153135
1	Feature 1	0.145160
4	Feature 4	0.095494
2	Feature 2	0.059639

Naive Bayes With Feature Selection (via Random Forest)
Accuracy Score: 0.7630057803468208

Classification Report:

	precision	recall	f1-score	support
0	0.92	0.92	0.92	250
1	0.56	0.31	0.40	71
2	0.29	0.14	0.19	14
3	0.22	1.00	0.36	11
accuracy			0.76	346
macro avg	0.50	0.59	0.47	346
weighted avg	0.80	0.76	0.76	346

The Naive Bayes model produced the lowest accuracy among all the algorithms tested, with a score of 76% both with and without feature selection. This result is expected, as the categorical features in the dataset likely violate the algorithm's independence assumptions. The model achieved a weighted F1-score of 0.76, indicating moderate overall performance, but the macro F1-score of 0.47 highlights significant struggles with minority class predictions. For instance, the model had very low recall for the 'good' class (0.14) and low F1-scores across all non-majority classes. As with other models, feature selection provided no improvement, suggesting that all features were relevant to the classification. While the feature importance scores aligned with other models, identifying safety as the most important at 0.302, this did not lead to competitive performance. Overall, Naive Bayes proved to be the least effective model for this multi-class classification problem.

Support Vector Machine (SVM) is a powerful supervised learning algorithm that works well in high-dimensional spaces and is effective for both linear and non-linear classification tasks. It is particularly useful for handling multi-class problems, which are handled by default using a one-vs-rest strategy in the SVC implementation from scikit-learn. In this project, we applied the SVM model to the full dataset and also tested a reduced feature set based on feature importance scores derived from the Random Forest classifier.

SVM Without Feature Selection
Accuracy Score: 0.9161849710982659

Classification Report:

	precision	recall	f1-score	support
0	0.97	0.98	0.97	250
1	0.78	0.85	0.81	71
2	0.50	0.36	0.42	14
3	1.00	0.73	0.84	11
accuracy			0.92	346
macro avg	0.81	0.73	0.76	346
weighted avg	0.91	0.92	0.91	346

Feature Importances:

	Feature	Importance
5	safety	0.302062
3	persons	0.244511
0	buy price	0.153135
1	maintain price	0.145160
4	luggage_storage	0.095494
2	doors	0.059639

SVM With Feature Selection (via Random Forest)
Accuracy Score: 0.9161849710982659

Classification Report:

	precision	recall	f1-score	support
0	0.97	0.98	0.97	250
1	0.78	0.85	0.81	71
2	0.50	0.36	0.42	14
3	1.00	0.73	0.84	11
accuracy			0.92	346
macro avg	0.81	0.73	0.76	346
weighted avg	0.91	0.92	0.91	346

The Support Vector Machine (SVM) model achieved an accuracy of 92% with and without feature selection, showing consistent performance regardless of the input feature set. Its weighted F1-score of 0.91 indicates strong overall effectiveness, while the macro F1-score of 0.76 reveals slightly weaker performance on minority classes. Similar to other models, feature selection provided no benefit, reinforcing that all attributes play a role in predicting the target variable. Although SVM outperformed Naive Bayes and Decision Tree in terms of both accuracy and F1-scores, it did not surpass the results of the Random Forest or KNN models. Overall, SVM proved to be a reliable and well-balanced classifier but not the most optimal for this dataset.

Algorithm	Feature Selection	Accuracy	Precision (Macro avg, Weighted avg)	Recall (Macro avg, Weighted avg)	F1-Score (Macro avg, Weighted avg)
Decision Tree	No	88%	0.54, 0.86	0.65, 0.88	0.59, 0.87
Decision Tree	Yes	88%	0.54, 0.86	0.65, 0.88	0.59, 0.87
Random Forest	No	98%	0.90, 0.98	0.85, 0.98	0.87, 0.98
Random Forest	Yes	98%	0.90, 0.98	0.85, 0.98	0.87, 0.98
KNN	No	95%	0.90, 0.95	0.83, 0.95	0.85, 0.95
KNN	Yes	95%	0.90, 0.95	0.83, 0.95	0.85, 0.95
Naive Bayes	No	76%	0.50, 0.80	0.59, 0.76	0.47, 0.76
Naive Bayes	Yes	76%	0.50, 0.80	0.59, 0.76	0.47, 0.76
SVM	No	92%	0.81, 0.91	0.73, 0.92	0.76, 0.91
SVM	Yes	92%	0.81, 0.91	0.73, 0.92	0.76, 0.91

We applied six different classification models: Decision Tree, Random Forest, K-Nearest Neighbors (KNN), Naive Bayes, and Support Vector Machines (SVM). Each model was evaluated both with and without feature selection to assess its impact on performance. Among the tested models, the Random Forest algorithm consistently performed the best, achieving an accuracy of 98%, along with the highest precision, recall, and F1-scores. This highlights the effectiveness of ensemble methods for categorical classification problems and their ability to minimize overfitting. While KNN and SVM also performed well with accuracies of 95% and 92%, they did not outperform Random Forest. Feature selection analysis showed that all features contribute meaningfully to the prediction, as the accuracy remained virtually unchanged whether feature selection was applied or not. This indicates that feature selection is not necessary for this dataset. However, the safety rating stood out as the most important feature influencing car acceptability, as identified by feature importance scores from the tree-based models. Overall, this project demonstrates the importance of trying multiple algorithms and assessing them using appropriate evaluation metrics. It also reinforces the value of proper preprocessing when handling categorical data.

References

Kanyi, E. (2024, February 24). *Car evaluation dataset*. Kaggle. https://www.kaggle.com/datasets/kanyianalyst/car-evaluation-dataset?resource=download