MOBILE PRIZE RANGE PREDICTION

AGENDA

- Data collection
- Dimensionality reduction
- Classification

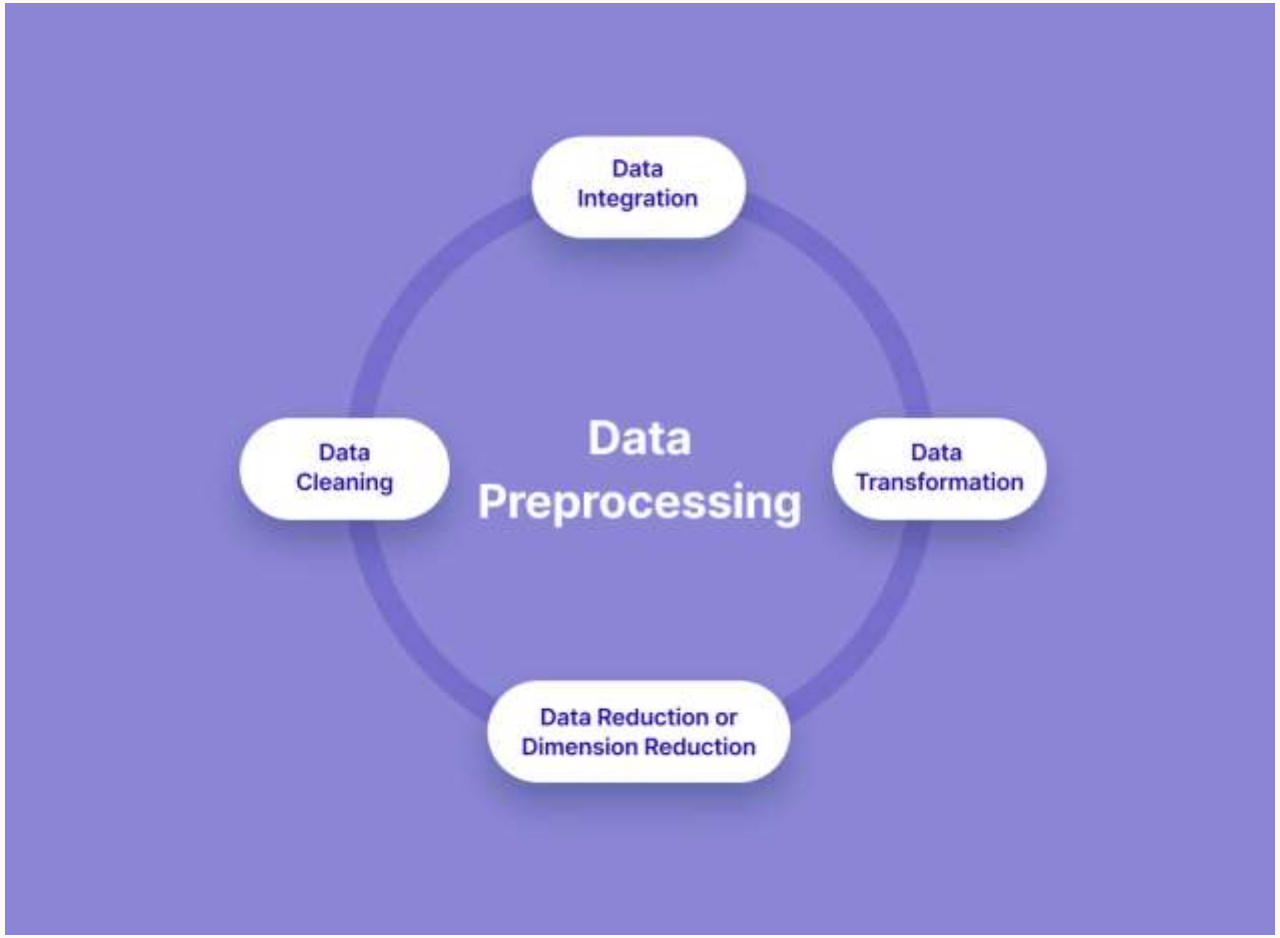
INTRODUCTION

today's hectic human existence. Size and thickness of the mobile device are other essential factors to consider when making a selection. Internal memory, camera pixels, and video consistency must all be recalled. Internet browsing is also one of the most important technical constraints of the twenty-first century. Also, the list of various features is determined by the size of the mobile device. As a result, we'll utilise all of the aforementioned characteristics to decide if the smartphone will be very-costly, economical, pricey, or very-costly. The following is a diagram of the paper's structure. The examination of past work is the next section. Technique andExperimental Procedure are covered in the third section. The results are described in Section 4 of the report. In section 5, a comparative analysis is carried out. After the section 6 paper is completed. The work's outcomes are discussed in section 7. Finally, in the eighth part, some recommendations for further research are made.

RELATED WORK

The use of prior data to estimate the pricing of available and new launch products is an intriguing study background for machine-learning researchers. Sameerchand-Pudaruth[1] estimates the prices of used automobiles in Mauritius. He used a variety of approaches to forecast prices, including multiple linear regressions, k-nearest neighbours (KNN), Decision Tree, and Nave Bayes. Sameerchand-Pudaruth obtained equivalent results using all of these approaches. During study, it was discovered that the majority of standard algorithms, such as Decision Tree and Nave Bayes, are incapable of processing, categorising, and forecasting numeric data.

METHODOLOGY



DATA COLLECTION

	battery_power	blue	clock_speed	dual_sim	fc	four_g	int_memory	m_dep	mobile_wt	n_cores	...	px_height
0	842	0	2.2	0	1	0	7	0.6	188	2	...	20
1	1021	1	0.5	1	0	1	53	0.7	136	3	...	905
2	563	1	0.5	1	2	1	41	0.9	145	5	...	1263
3	615	1	2.5	0	0	0	10	0.8	131	6	...	1216
4	1821	1	1.2	0	13	1	44	0.6	141	2	...	1208

DIMENSIONALITY REDUCTION

he technique of decreasing the number of random variables (Features) under consideration by getting a set of main variables is known as dimensionality reduction[7]. The more characteristics there are, the more difficult it is to envision the training set and subsequently work on it. Most of these characteristics are sometimes linked and therefore redundant. Dimensionality reduction techniques are useful in this situation[7]. There are two types of dimension reduction algorithms: feature selection and feature extraction

CLASSIFICATION

LET'S GO ON TO THE FINAL STAGE, CLASSIFICATION. AS PREVIOUSLY STATED, A SEPARATE TEST SET IS UTILISED TO ASSESS THE CLASSIFIER AND DETERMINE ACCURACY. ANY CLASSIFICATION IS CORRECT IF THE NUMBER OF CORRECTLY IDENTIFIED CLASS SAMPLES (TRUE POSITIVES), CORRECTLY IDENTIFIED SAMPLES THAT ARE NOT MEMBERS OF THE CLASS (TRUE NEGATIVES), AND SAMPLES THAT WERE EITHER INCORRECTLY ASSIGNED TO THE CLASS (FALSE POSITIVES) OR NOT IDENTIFIED AS CLASS SAMPLES (FALSE NEGATIVES) CAN BE CALCULATED[10]. THE PERCENTAGE OF ACCURATELY CATEGORISED CASES IS CALLED ACCURACY. MATHEMATICALLY

SUMMARY

At Contoso, we believe in giving 110%. By using our next-generation data architecture, we help organizations virtually manage agile workflows. We thrive because of our market knowledge and great team behind our product. As our CEO says, "Efficiencies will come from proactively transforming how we do business."

ACCURACY

$$\text{Accuracy} = \frac{(TP + TN)}{(TP + FP + TN + FN)}$$

```
In [124]: import pandas as pd
          import matplotlib.pyplot as plt
          import seaborn as sns

In [125]: train=pd.read_csv(r'train.csv')
          test=pd.read_csv(r'test.csv')

In [126]: pd.set_option('display.max_rows',None)
          pd.set_option('display.max_columns',None)
```

[127]: `train.head()`

Out[127]:

	battery_power	blue	clock_speed	dual_sim	fc	four_g	int_memory	m_dep	mobile_wt	n_cores	pc	px_height	px_width	ram	sc_h	sc_w	talk_time	th
0	842	0	2.2	0	1	0	7	0.6	188	2	2	20	756	2549	9	7	19	
1	1021	1	0.5	1	0	1	53	0.7	136	3	6	905	1988	2631	17	3	7	
2	563	1	0.5	1	2	1	41	0.9	145	5	6	1263	1716	2603	11	2	9	
3	615	1	2.5	0	0	0	10	0.8	131	6	9	1216	1786	2769	16	8	11	
4	1821	1	1.2	0	13	1	44	0.6	141	2	14	1208	1212	1411	8	2	15	

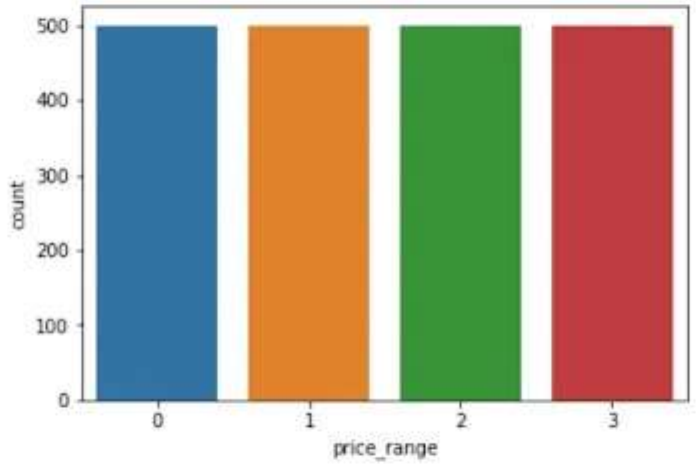
[128]: `test.head()`

Out[128]:

	id	battery_power	blue	clock_speed	dual_sim	fc	four_g	int_memory	m_dep	mobile_wt	n_cores	pc	px_height	px_width	ram	sc_h	sc_w	talk_time
0	1	1043	1	1.8	1	14	0	5	0.1	193	3	16	226	1412	3476	12	7	2
1	2	841	1	0.5	1	4	1	61	0.8	191	5	12	746	857	3895	6	0	7
2	3	1807	1	2.8	0	1	0	27	0.9	186	3	4	1270	1366	2396	17	10	10
3	4	1546	0	0.5	1	18	1	25	0.5	96	8	20	295	1752	3893	10	0	7
4	5	1434	0	1.4	0	11	1	49	0.5	108	6	18	749	810	1773	15	8	7

[129]: `test.drop('id',axis=1,inplace=True)`

```
131]: sns.countplot(train['price_range'])

131]: <matplotlib.axes._subplots.AxesSubplot at 0x25277992ba8>
```



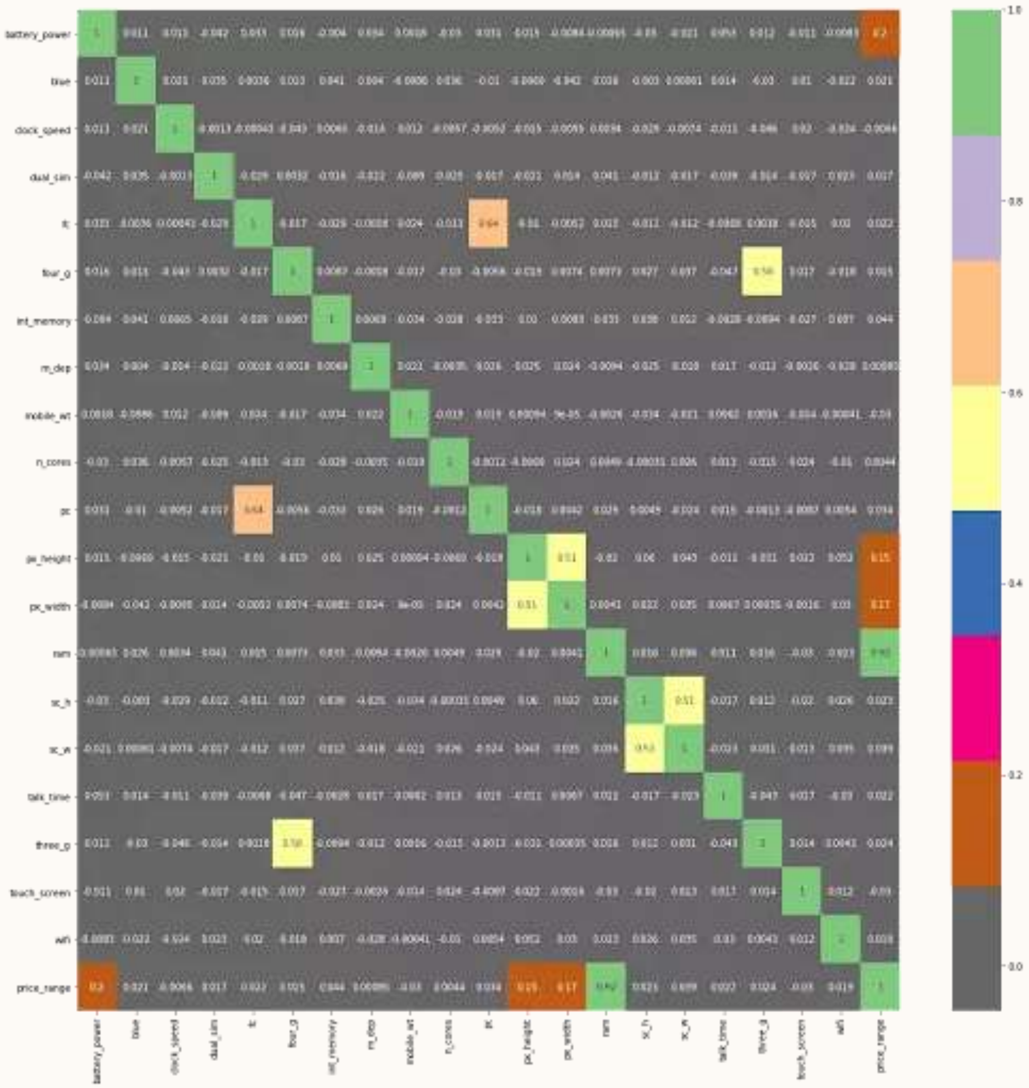

```
132]: train.shape,test.shape

132]: ((2000, 21), (1000, 20))
```

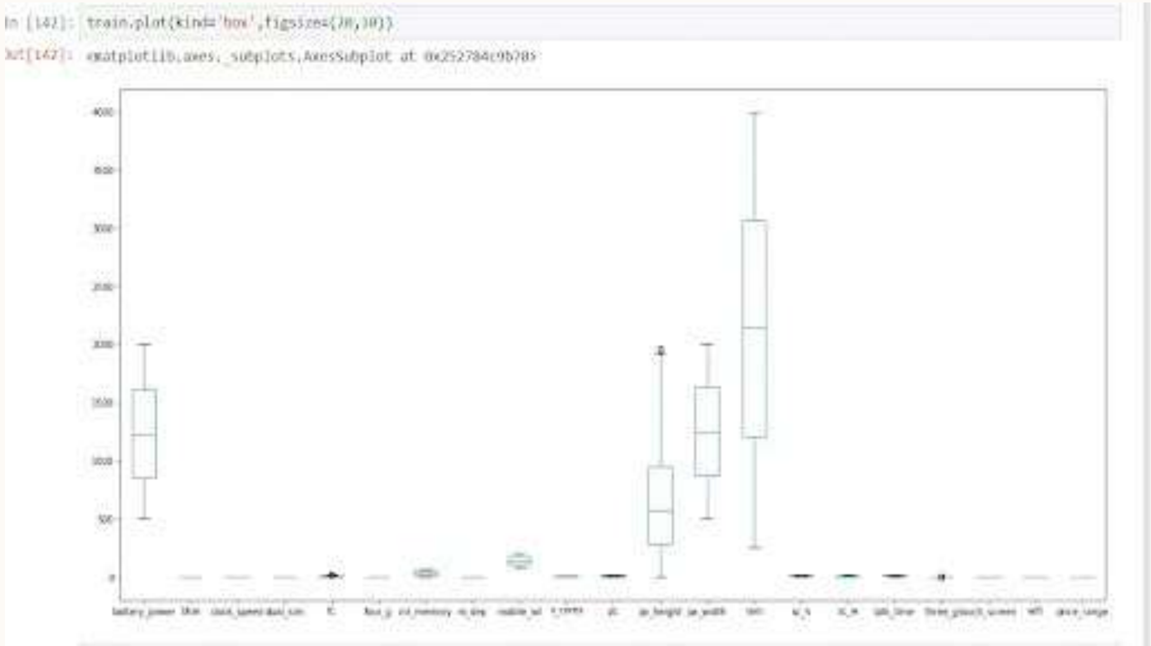
```
n [133]: train.isnull().sum()
```

```
ut[133]: battery_power    0
         blue             0
         clock_speed      0
         dual_sim         0
         fc               0
         four_g           0
         int_memory       0
         m_dep            0
         mobile_wt        0
         n_cores          0
         pc               0
         px_height        0
         px_width         0
         ram              0
         sc_h             0
         sc_w             0
         talk_time        0
         three_g          0
         touch_screen     0
```

```
n [134]: train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2000 entries, 0 to 1999
Data columns (total 21 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   battery_power  2000 non-null   int64  
 1   blue           2000 non-null   int64  
 2   clock_speed    2000 non-null   float64
 3   dual_sim       2000 non-null   int64  
 4   fc             2000 non-null   int64  
 5   four_g         2000 non-null   int64  
 6   int_memory     2000 non-null   int64  
 7   m_dep          2000 non-null   float64
 8   mobile_wt      2000 non-null   int64  
 9   n_cores        2000 non-null   int64  
```


battery_power
blue
clock_speed
dual_sim
fc
four_g
int_memory
m_dep
mobile_wt
n_cores
pc
px_height
px_width
ram
sc_h
sc_w
talk_time
three_g
touch_screen
wifi
price_range
1.0
0.8
0.6
0.4
0.2
0.0



Then check the outliers in the dataset. But no outliers are present.

```
[143]: X = train.drop('price_range',axis=1)
       y = train['price_range']

[145]: from sklearn.model_selection import train_test_split
       X_train, X_test, Y_train, Y_test = train_test_split(X,y,test_size=0.1,random_state=101)

[146]: from sklearn.preprocessing import StandardScaler
       sc = StandardScaler()
       X_train = sc.fit_transform(X_train)
       X_test = sc.transform(X_test)
       test = sc.transform(test)
```

Now split the dataset into the independent and dependent features.
Then split the dataset into the training and testing to evaluate the model using the train_test_split method.
Then apply the standardization on the training and testing datasets. Standardization makes all the features' value in a particular range (0-1).

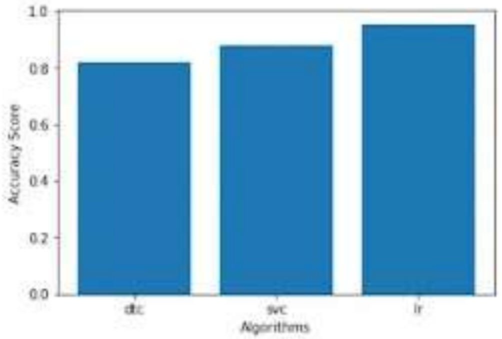
```
164]: from sklearn.tree import DecisionTreeClassifier
      dtc = DecisionTreeClassifier()
      dtc.fit(X_train , Y_train)

164]: DecisionTreeClassifier()

165]: pred = dtc.predict(X_test)
      pred

165]: array([1, 1, 2, 1, 0, 1, 2, 1, 1, 1, 0, 1, 2, 1, 1, 0, 1, 1, 1, 0, 3, 1,
             2, 3, 2, 2, 2, 1, 0, 0, 2, 3, 0, 0, 3, 0, 0, 0, 1, 1, 1, 2, 3, 2,
             2, 1, 1, 3, 3, 1, 0, 0, 2, 3, 3, 2, 0, 3, 2, 3, 2, 2, 3, 1, 3, 2,
             0, 1, 0, 2, 1, 2, 3, 2, 1, 3, 3, 2, 0, 2, 0, 0, 2, 1, 2, 2, 2, 1,
             0, 0, 3, 3, 0, 2, 0, 3, 2, 0, 2, 3, 0, 1, 2, 3, 0, 2, 0, 0, 2, 0,
             1, 0, 3, 2, 2, 2, 1, 3, 2, 0, 3, 3, 2, 3, 1, 3, 3, 2, 1, 1, 0, 0,
             1, 1, 0, 2, 3, 0, 2, 3, 1, 3, 0, 1, 0, 0, 1, 3, 3, 0, 2, 1, 3, 2,
             3, 3, 2, 0, 3, 1, 2, 2, 2, 2, 1, 2, 1, 1, 3, 3, 1, 2, 0, 3, 2, 3,
             1, 2, 3, 1, 2, 1, 0, 1, 3, 3, 1, 2, 1, 3, 1, 0, 2, 2, 0, 3, 0, 0,
             3, 0], dtype=int64)

167]: from sklearn.metrics import accuracy_score, confusion_matrix
      dtc_acc = accuracy_score(pred,Y_test)
      print(dtc_acc)
      print(confusion_matrix(pred,Y_test))

      0.83
      [[43  4  0  0]
       [ 7 37  7  0]
       [ 0  5 47  3]
       [ 0  0  8 39]]
```

Now load the Decision Tree Classifier from sklearn library and define the DecisionTreeClassifier and train with the X_train and Y_train dataset. Then test the model using the X_test dataset. Then check the accuracy score of the Decision Tree Classifier. As you can see, the accuracy score is approx 83%.

```
[156]: from sklearn.linear_model import LogisticRegression  # its a classification
       lr=LogisticRegression()
       lr.fit(X_train,Y_train)

t[156]: LogisticRegression()

[157]: pred2 = lr.predict(X_test)
       pred2

t[157]: array([1, 1, 2, 1, 1, 1, 2, 1, 1, 1, 0, 1, 1, 1, 1, 0, 0, 1, 0, 0, 3, 1,
               2, 3, 2, 2, 2, 2, 0, 0, 2, 3, 0, 0, 3, 0, 0, 0, 1, 1, 1, 2, 3, 2,
               3, 0, 1, 3, 3, 1, 0, 0, 3, 3, 3, 3, 1, 3, 2, 3, 2, 2, 3, 1, 3, 1,
               0, 0, 0, 2, 1, 2, 3, 2, 1, 3, 3, 2, 0, 2, 0, 0, 2, 1, 2, 2, 2, 1,
               0, 0, 3, 2, 0, 2, 0, 3, 2, 0, 2, 3, 0, 1, 3, 3, 0, 3, 0, 0, 2, 0,
               1, 0, 3, 2, 2, 1, 1, 3, 1, 0, 3, 2, 2, 3, 1, 2, 3, 2, 1, 1, 1, 0,
               0, 1, 0, 2, 3, 0, 2, 3, 1, 3, 0, 0, 0, 1, 1, 2, 2, 0, 3, 1, 2, 2,
               3, 2, 2, 0, 3, 2, 2, 2, 2, 2, 1, 2, 1, 1, 3, 3, 1, 2, 0, 3, 1, 3,
               2, 2, 3, 2, 2, 1, 0, 1, 3, 2, 1, 2, 0, 3, 1, 0, 2, 2, 0, 2, 0, 0,
               3, 0], dtype=int64)

[170]: from sklearn.metrics import accuracy_score
       lr_acc = accuracy_score(pred2,Y_test)
       print(lr_acc)
       print(confusion_matrix(pred2,Y_test))

0.955
[[49  1  0  0]
 [ 1 45  3  0]
 [ 0  0 56  1]
 [ 0  0  3 41]]
```

Now load the Logistic Regression and define the LogisticRegression and train with the X_train and Y_train dataset. Then test the model using the X_test dataset. Then check the accuracy score of the Logistic Regression. As you can see, the accuracy score is approx 95%.

```
[159]: plt.bar(x=['dtc','svc','lr'],height=[dtc_acc,svc_acc,lr_acc])
       plt.xlabel("Algorithms")
       plt.ylabel("Accuracy Score")
       plt.show()
```




```
[160]: lr.predict(test)
```

```
[160]: array([3, 3, 2, 3, 1, 3, 3, 1, 3, 0, 3, 3, 0, 0, 2, 0, 2, 1, 3, 2, 1, 3,
              1, 1, 3, 0, 2, 0, 3, 0, 2, 0, 3, 0, 1, 1, 3, 1, 2, 1, 1, 2, 0, 0,
              0, 1, 0, 3, 1, 2, 1, 0, 3, 0, 3, 0, 3, 1, 1, 3, 3, 3, 0, 1, 1, 1,
              2, 3, 1, 2, 1, 2, 2, 3, 3, 0, 2, 0, 1, 3, 0, 3, 3, 0, 3, 0, 3, 1,
              3, 0, 1, 2, 2, 1, 2, 2, 0, 2, 1, 2, 1, 0, 0, 3, 0, 2, 1, 1, 2, 3,
              3, 3, 1, 3, 3, 3, 3, 2, 3, 0, 0, 3, 2, 1, 2, 0, 3, 2, 2, 2, 0, 2,
              2, 1, 3, 1, 1, 0, 3, 2, 1, 2, 1, 3, 2, 3, 3, 3, 2, 3, 2, 3, 1, 0,
              3, 2, 3, 3, 3, 3, 3, 2, 3, 3, 3, 3, 1, 0, 3, 0, 0, 0, 2, 1, 0, 1,
              0, 0, 1, 2, 1, 0, 0, 1, 1, 2, 2, 1, 0, 0, 0, 1, 0, 3, 1, 0, 2, 2,
              3, 3, 1, 2, 3, 2, 3, 2, 2, 1, 0, 0, 1, 3, 0, 2, 3, 3, 0, 2, 0, 3,
              2, 3, 3, 1, 0, 1, 0, 3, 0, 1, 0, 2, 2, 1, 3, 1, 3, 0, 3, 1, 2, 0,
              0, 2, 1, 3, 3, 3, 1, 1, 3, 0, 0, 2, 3, 3, 1, 3, 1, 1, 3, 2, 1, 2,
```

CONCLUSION

we looked at classification. Classifiers represent the intersection of advanced machine theory and practical application. These algorithms are more than just a sorting mechanism for organising unlabeled data instances into distinct groupings. Classifiers include a unique set of dynamic rules that include an interpretation mechanism for dealing with ambiguous or unknown values, all of which are suited to the kind of inputs being analysed. Most classifiers also utilise probability estimates, which enable end-users to adjust data categorization using utility functions.