CITS3401 - Data Warehousing - Project 2: Pattern Discovery and Building Predictive Models of Mobile Phone Price

Max Matthews (21506225), Frinze Erin Lapuz (22711649)

May 21, 2021

1 Introduction

For this project, we would like to use the mobile price classification dataset as the source of data. The target of this project is to predict whether the price of a mobile phone is high or not.

1.1 Tasks and Scope

1.1.1 1) Data cleaning and analysis

- Read through the table and the table column descriptions. Understand the meaning of each column in the table.
- Distinguish the type of each attribute (e.g., nominal/categorical, numerical). You may need to discretise some attributes, when completing Task 2, 3 or 4.
- Determine whether an attribute is relevant to your target variable. You may remove some attributes if they are not helpful for Task 2, 3, or 4. You might create separate data files for Task 2, 3 and 4.
- Identify inconsistent data and take actions using the knowledge you have learnt in this unit.

1.1.2 2) Association rule mining

- Select a subset of the attributes (or all the attributes) to mine interesting patterns. To rank the degree of interesting of the rules extracted, use support, confidence and lift.
- Explain the top k rules (according to lift or confidence) that have the "price_category" on the right-hand-side, where k >= 1.
- Explain the meaning of the k rules in plain English.
- Given the rules, what recommendation will you give to a company willing to design a high price mobile phone (e.g., should the mobile phone equipped with bluetooth)?

1.1.3 3) Classification

- Use the "price_category" as the target variable and train two classifiers based on different machine learning algorithms (e.g. classifier 1 based on a decision tree; classifier 2 based on SVMs).
- Evaluate the classifiers based on some evaluation metrics (e.g., accuracy). You may use 10-fold cross-validation for the evaluation.

1.1.4 4) Clustering

- Run a clustering algorithm of your choice and explain how the results can be interpreted with respect to the target variable.

1.1.5 5) Data reduction

- Perform numerosity reduction and perform attribute reduction.
- Train the two classifiers in Task 3 on the reduced data.
- Answer the question: "Does data reduction improve the quality of the classifiers"?

1.1.6 6) Attribute selection

- Select the top-10 most important attributes manually based on your understanding of the problem; select the top-10 most important attributes based on Information Gain.
- Which attribute selection method is better and why?

1.2 Marking Scheme

[5 marks] Explain the data processing operations (e.g., remove some attributes and action on inconsistent data) that you have done.

[5 marks] Explain and interpret the top k association rules mined; based on the association rules, provide a recommendation for a company willing to design a high price mobile phone.

[5 marks] Explain how you train the classifiers and your evaluation results.

[5 marks] Clustering and interpretation of the clustering result (with respect to the target variable).

[5 marks] Explain the data reduction you have performed; compare the classifiers trained on reduced data with the classifiers trained on the original data.

[5 marks] Your answer to Task 6.

1.3 Tools, Libraries and Packages

Python - Used throughout the project for data cleaning, data processing, and modelling.

1.3.1 Imports

https://graphviz.org/download/

[ ]:
```
pip install pandas-profiling[notebook] mlxtend clusteval pca graphviz dtreeviz␣
↪tabulate --user
```

[2]:
```
# Data analysis, manipulation, and profiling
import pandas as pd
pd.options.display.max_colwidth = 100
pd.set_option("display.notebook_repr_html", False)

import numpy as np
from pandas_profiling import ProfileReport
```

```
# Visualization
import matplotlib.pyplot as plt
import seaborn as sns
sns.set_style("darkgrid", {"axes.facecolor": ".9"})

# Association Rule Mining
from mlxtend.frequent_patterns import apriori, association_rules

# Training Setups
from sklearn.model_selection import train_test_split
from sklearn.pipeline import Pipeline

# Preprocessings and Attribute Selections
from sklearn.impute import SimpleImputer
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
# selection of best attributes from by default using f-score https://
 ↪scikit-learn.org/stable/modules/generated/sklearn.feature_selection.
 ↪f_classif.html
from sklearn.feature_selection import SelectKBest, mutual_info_classif,␣
 ↪f_classif

# Classifiers
from sklearn.tree import DecisionTreeClassifier, plot_tree
from sklearn.svm import SVC # Support Vector Machine Classifier
from sklearn import tree
import graphviz
from dtreeviz.trees import dtreeviz

RANDOM_STATE = 1 # Used as a seed value
CROSS_VALIDATION_PARTITION = 10
NUMBER_OF_BEST_HYPER_PARAMS_TO_SHOW = 5

# Optimisation
from sklearn.model_selection import GridSearchCV # tuning the model
from sklearn.model_selection import cross_val_score, cross_validate

# Clustering
from clusteval import clusteval
from sklearn.preprocessing import MinMaxScaler
from sklearn.cluster import KMeans, AgglomerativeClustering
from scipy.cluster import hierarchy
from scipy.cluster.hierarchy import dendrogram

# Data Reduction
from sklearn.ensemble import IsolationForest
```

```
from sklearn.svm import OneClassSVM
from sklearn.covariance import EllipticEnvelope
from sklearn.cluster import DBSCAN
from pca import pca
```

2 Data Cleaning and Profiling

There are many ways to approach a project's data cleaning, and data profiling steps. For these processes (in Project 2) we will be using an IPython Notebook, for the following reasons:

- Both group members are proficient in Python;
- The report can be integrated with code for specific sections of the analysis;
- The processes / procedures are highly repeatable and easily automated using scripts;
- Data exploration and anomaly detection can easily be performed through a variety of visualizations (charts, graphs, tables, etc);

The packages that will be used are built in to the default Anaconda package, with exception to `pandas_profiling`, `mlxtend`, and `pca`.

- `pandas_profiling` (from https://github.com/pandas-profiling/pandas-profiling) was leveraged to provide a detailed exploratory analysis of our data, and the attributes we would be working with. Data profiling is crucial to the measurement of the quality of data, which in turn greatly assists the analyst team in their discovery of data anomalies/inconsistencies, and as such, appropriate data transformations and/or pre-processing actions. The data-profiling reports (generated by this package) will be referenced in our project discussion, with the full resource located in the appendix.
- `mlxtend` (from https://github.com/rasbt/mlxtend) was used in our Association Rule Mining section, to assist in the mining of such associated rules.
- `pca` (from https://github.com/erdogant/pca) was used for graphical presentation of the Principal Component Analysis.

2.1 Profiling Report

With the imported metadata, and raw dataset, we can now populate and generate a `pandas_profiling` report. For reference, see `Raw Data Profiling Report`, in the apendix.

2.2 Raw Data - Interpretation, Analysis, and Cleaning

Here, we will walk through the fields presented in the raw data, stating assumptions/decisions, and undertaking any appropriate cleaning steps. In the event of changes, we will apply them to the `staging_data` data frame, and `staging_metadata` dictionary.

`ID` The id field is unique, and it seems to be a good candidate for primary key. Note, we will NOT be using ID for any analysis, as it is an irrelevant attribute.

---

battery_power The battery_power has 1094 distinct counts, approximately half the distinct counts of id. This data is continuous, and as such, we shall discretise for the later association rule mining.

```
[ ]: staging_data = create_discretised_col(staging_data, "battery_power",␣
     ↪is_zero_its_own_category = False)
```

---

blue This ("has bluetooth") field is a categorical data type (*boolean*), expressed (poorly) in string-form, with many inconsistencies. We observe 10 distinct values that all refer to either the phone having bluetooth (True), or not having bluetooth (False). The inconsistencies must be attributed to an appropriate boolean value, and as such, this field needs to be cleaned.

```
[ ]: staging_data["has_bluetooth"] = staging_data["blue"].apply(string_to_bool)
     staging_metadata.loc["blue","new_name"] = "has_bluetooth"  # Rename metadata
```

---

clock_speed The clock_speed attribute has 26 distinct values for which 413 of the records have a value of 0.5. We note that the distribution of values in the histogram is right skewed. This data is continuous and will need to be discretised if we wish to use it in the association rule mining process.

```
[ ]: staging_data = create_discretised_col(staging_data, "clock_speed",␣
     ↪is_zero_its_own_category = False)
```

---

dual_sim The dual_sim attirbute is similar to the **blue** / **has_bluetooth** attribute, wherein it is a categorical datatype (*boolean*) with its inconsistencies forming 10 (*string*) distinct values. Hence, the procedure of cleaning will be similar.

```
[ ]: staging_data["has_dual_sim"] = staging_data["dual_sim"].apply(string_to_bool)
     staging_metadata.loc["dual_sim","new_name"] = "has_dual_sim"  # Rename metadata
```

---

fc The (*Front Camera*) **fc** attribute has 474 (23.7% of the dataset) zero values. We will be interpreting the zero values as "this phone does not have a front camera" (please see the ***Data Privacy Disclaimer*** below). This continous data will be discretised for later (association rule mining) purposes, with present zero values (as a result of our domain interpretation) populating their own level within this category.

```
[ ]: staging_data["front_cam_resolution"] = staging_data["fc"]
     staging_metadata.loc["fc","new_name"] = "front_cam_resolution" # Rename metadata
     staging_data = create_discretised_col(staging_data, "front_cam_resolution",␣
     ↪is_zero_its_own_category = True)
```

---

four_g The four_g attribute is a categorical (*boolean*) attribute, with values indicated as yes = 1 and no = 0 regarding (the phone's) 4G capability. For consistency, this will be converted from current numeric form to boolean type.

[ ]:
```
staging_data["has_four_g"] = staging_data["four_g"].apply(int_to_bool)
staging_metadata.loc["four_g","new_name"] = "has_four_g" # Rename metadata
```

---

int_memory The int_memory (*Internal Memory*) attribute has 63 distinct values that vary as shown in the *profiling-report* histogram. There are no records of phones having 0 GB as the storage which would be expected, as phones would require a minimum amount of memory to host their respective operating software/s. This continuous data will be discretised for later use in the association rule mining phase.

[ ]:
```
staging_data = create_discretised_col(staging_data, "int_memory",␣
↪is_zero_its_own_category = False)
```

---

m_dep The m_dep (*Mobile Depth*) attribute has 10 distinct values that vary as shown in the *profiling-report* histogram. The minimum values do not make much sense, however, please refer to the ***Data Privacy Disclaimer*** below. Additionally, this continuous data will be discretised for later (association rule mining) purposes.

[ ]:
```
staging_data["mobile_depth"] = staging_data["m_dep"]
staging_metadata.loc["m_dep","new_name"] = "mobile_depth"  # Rename metadata
staging_data = create_discretised_col(staging_data, "mobile_depth",␣
↪is_zero_its_own_category = False)
```

---

mobile_wt The mobile_wt (*Mobile Weight*) attribute has 121 distinct values that vary as shown in the *profiling-report* histogram. The respective min and max values are 80 and 200, measured in presumably grams. This continuous data will be discritised for later association rule mining purposes.

[ ]:
```
staging_data["mobile_weight"] = staging_data["mobile_wt"]
staging_metadata.loc["mobile_wt","new_name"] = "mobile_weight" # Rename metadata
staging_data = create_discretised_col(staging_data, "mobile_weight",␣
↪is_zero_its_own_category = False)
```

---

n_cores The n_cores (*Number of Cores*) attribute looks almost uniform in the range 1-8. This is categorical (numerical) data, and requires no further attention at this moment.

[ ]:
```
staging_data["number_of_cores"] = staging_data["n_cores"]
staging_metadata.loc["n_cores","new_name"] = "number_of_cores" # Rename metadata
```

---

pc The pc (*Primary Camera Resolution*) attribute displays 21 distinct values, with 101 zero values (5.1% of the data). We will be interpreting the zero values as "this phone does not have a primary camera", (please see the ***Data Privacy Disclaimer*** below). This continuous data will be discretised for later (association rule mining) purposes, with present zero values (as a result of our domain interpretation) populating their own level within this category.

[ ]:
```
staging_data["primary_cam_resolution"] = staging_data["pc"]
staging_metadata.loc["pc","new_name"] = "primary_cam_resolution"
staging_data = create_discretised_col(staging_data, "primary_cam_resolution",␣
↪is_zero_its_own_category = True)
```

---

px_height The px_height (*Pixel Height*) attribute has a right-skewed, normal distribution, with a mean and standard deviation of 645 and 443.79 pixels (respectively), and values falling in the range 0-1960. We will be interpreting the two zero values as *extremely small, but not zero* values, (please see the ***Data Privacy Disclaimer*** below). This notion also applies to the nonsensical low values. For association rule mining purposes, this continuous data will be discretised, wherein (as per the assumption above) the zero values will fall within the first category, shared by other values (relatively) close to zero.

[19]:
```
staging_data = create_discretised_col(staging_data, "px_height",␣
↪is_zero_its_own_category = False)
```

---

px_width The px_width (*Pixel Width*) attribute has a varying distribution of values in the range 500 to 1998. This continuous data will be discretised, for later (association rule mining) purposes.

[ ]:
```
staging_data = create_discretised_col(staging_data, "px_width",␣
↪is_zero_its_own_category = False)
```

---

ram The ram attribute has a varying distribution of values in the range 256-3998. This continuous data will be discretised, for later (association rule mining) purposes.

[ ]:
```
# Discretizing ram
staging_data = create_discretised_col(staging_data, "ram",␣
↪is_zero_its_own_category = False)
```

---

sc_h The sc_h (*Screen Height*) attributes has 15 distinct values with a range of 5-19. This continuous data will be discretised, for later (association rule mining) purposes.

```
staging_data["screen_height"] = staging_data["sc_h"]
staging_metadata.loc["sc_h","new_name"] = "screen_height" # Rename metadata
staging_data = create_discretised_col(staging_data, "screen_height",␣
↪is_zero_its_own_category = False)
```

---

sc_w The sc_w (*Screen Width*) attribute has 19 distinct values with a range of 0-18 with 180 zero values. These zero values do not make *real world sense*, and so we will be interpreting any zero values as representatives of sensible, low values (please see the ***Data Privacy Disclaimer*** below). Note that this notion also applies to any non-zero, nonsensical low values. For association rule mining purposes, this continuous data will be discretised, wherein (as per the assumption above) the zero values will fall within the first category, shared by other values (relatively) close to zero.

```
staging_data["screen_width"] = staging_data["sc_w"]
staging_metadata.loc["sc_w","new_name"] = "screen_width" # Rename metadata
staging_data = create_discretised_col(staging_data, "screen_width",␣
↪is_zero_its_own_category = False)
```

---

talk_time The talk_time attribute has 19 distinct values with a range 2-20. This continuous data will be discretised, for later (association rule mining) purposes.

```
staging_data = create_discretised_col(staging_data, "talk_time",␣
↪is_zero_its_own_category = False)
```

---

three_g The three_g attribute is similar to the has_bluetooth and dual_sim attribute, wherein it is a categorical (*boolean*) datatype, with inconsistencies spread to 10 (string) distinct values. As such, the cleaning procedure will be similar.

```
staging_data["has_three_g"] = staging_data["three_g"].apply(string_to_bool)
staging_metadata.loc["three_g","new_name"] = "has_three_g" # Rename metadata
```

---

touch_screen The touch_screen attribute is a categorical (*boolean*) attribute, similar to the initial state of the imported four_g attribute. Hence, the cleaning procedure is similar.

```
staging_data["has_touch_screen"] = staging_data["touch_screen"].
↪apply(int_to_bool)
staging_metadata.loc["touch_screen","new_name"] = "has_touch_screen" # Rename␣
↪metadata
```

---

wifi The `wifi` (*has wifi*) attribute is similar to `blue`,`dual_sim` and `three_g` attribute, wherein it is a boolean datatype with inconsistencies spread to 10 (string) distinct values. Hence, the procedure of cleaning will be similar.

[ ]:
```
staging_data["has_wifi"] = staging_data["wifi"].apply(string_to_bool)
staging_metadata.loc["wifi","new_name"] = "has_wifi" # Rename metadata
```

---

price_category The `price_category` attribute is categorical (*boolean*) data, similar to the representation of `four_g` and `touch_screen`. Hence, the cleaning procedure is similar.

[ ]:
```
staging_data["is_expensive"] = staging_data["price_category"].apply(int_to_bool)
staging_metadata.loc["price_category","new_name"] = "is_expensive" # Rename␣
↪metadata
```

2.3 *Data Privacy Disclaimer*

The data provided (`mobile_prices.csv`) has had some of it's *real world* values altered for privacy and/or legal reasons. As such, decisions were made on how to best interpret unusual / nonsensical values equal, or approximately equal to `zero`.

2.3.1 Appropriate zero values

- `primary_camera_resolution`
- `front_camera_resolution`

Zero values for these attributes can be observed as the phone not having the attribute. This makes sense, as not all phones have a front camera, or primary camera.

2.3.2 Inappropriate zero values

- `px_height`
- `screen_width`

Zero / close to zero values for these attributes can be observed as **values altered for privacy reasons** and should be interpreted as a representation of a relatively low value (but **NOT** taken literally as the stated value).

2.4 Set `cleaned_data` and `discretised_data`

- Declare our `cleaned_data`, `discretised_data` and `cleaned_metadata`;
- Generate "clean" pandas-profiling report;
- Export these to `.csv` files;

[ ]:
```
cleaned_data = staging_data[[
    "id","battery_power","has_bluetooth","clock_speed",
    "has_dual_sim","front_cam_resolution","has_four_g",
    "int_memory","mobile_depth","mobile_weight",
```

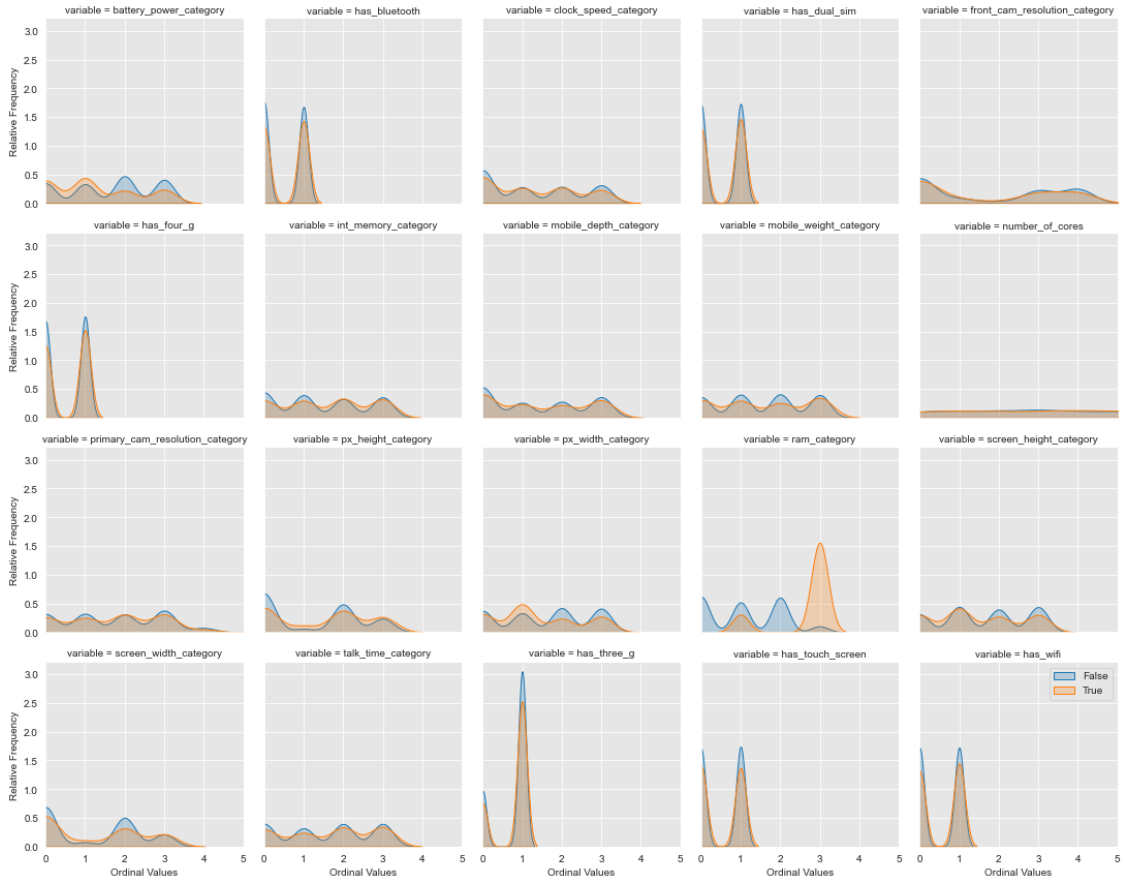
```
    "number_of_cores", "primary_cam_resolution", "px_height",
    "px_width", "ram", "screen_height","screen_width", "talk_time",
    "has_three_g","has_touch_screen", "has_wifi", "is_expensive"]]

discretised_data = staging_data[[
    "battery_power_category","has_bluetooth","clock_speed_category",
    "has_dual_sim","front_cam_resolution_category", "has_four_g",
    "int_memory_category","mobile_depth_category","mobile_weight_category",
    "number_of_cores", "primary_cam_resolution_category", "px_height_category",
    "px_width_category", "ram_category", "screen_height_category",
    "screen_width_category", "talk_time_category", "has_three_g",
    "has_touch_screen", "has_wifi", "is_expensive"]]
```

2.4.1 Pandas-Profiling (`cleaned_data` and `discretised_data`) Reports

See `Clean Data Profiling Report`, and `Discretised Data Profiling Report` (in appendix).

[ ]:
```
# Sort all the columns based on value (this will determine the Ordinal Encoder)

discretised_ordered_data = discretised_data.copy()

for column in discretised_data.columns:
    if column != 'is_expensive':
        unique_sorted_values =np.sort(np.unique(discretised_data[column].
 ↪astype(str)))
        dictionary = {}
        for numeric_value in range(len(unique_sorted_values)):
            dictionary[unique_sorted_values[numeric_value]] = numeric_value
        discretised_ordered_data[column] = discretised_data[column].
 ↪apply(lambda value: dictionary[str(value)])

discretised_ordered_data
```

[33]:
```
melted_data = pd.melt(discretised_ordered_data, "is_expensive",␣
 ↪discretised_ordered_data.drop(["is_expensive"],axis="columns"),␣
 ↪value_name="Ordinal Value",)

g = sns.FacetGrid(melted_data, col="variable", hue="is_expensive", col_wrap=5)
g.map(sns.kdeplot, "Ordinal Value", shade=True)
g.set_axis_labels("Ordinal Values", "Relative Frequency")

plt.xlim(0,5)
plt.legend()
plt.show()
```



This (above) shows the relative frequency graph of the discretised data, in ordinal form, split by the phone-data's `is_expensive` field. The variable `ram_category` displays a strong variance regarding its seperation with `is_expensive`, which following steps will investigate further.

3 Association Rule Mining

3.1 Information

Association rule mining ('ARM') is a technique used investigate and uncover frequently occurring patterns, correlations, or associations, between the characteristics of a given dataset/s or source. Association rules comprise of two parts:

- antecedent (if) → An antecedent is something that is found in data;
- consequent (then) → A consequent is an item that is found in combination with the antecedent.

3.2 ARM Approach

Our approach is to take the characteristics of each of the phone attributes (from our data) with the following strategy:

- keep the (current) cleaned data, especially the boolean values;
- discretise any numeric values;

Before the "apriori" runs, the data should be in a form similar to:

has_bluetooth	has_dual_sim	battery_power_0-100	...
True	False	True	True
True	True	False	True
False	True	True	False

For completed pre-processing, see IPython Notebook or `./data/cleaned/ARM_item_characteristics.csv`

3.3 Apriori Algorithm

Apriori (*Apriori Algorithm*) is an algorithm used in frequent-item-set / pattern mining over relational datasets. It seeks to identify frequent, individual items in the data, and then extend them to (incrementally) larger item sets, on the proviso that such item sets appear sufficiently often enough in said data.

3.4 Support

The recognition of frequently occurring patterns (in a dataset) must first be tuned by certain analyst-defined parameters. The minimum support acts as an 'entry requirement value' that rules must meet/exceed, for them to be considered 'frequent'. A minimum support value set too low, will not filter out patterns. If it is set too high, this strict program's mining expedition will return no rules, or only extremely dominant attribute-patterns. Example below, where we set `MINIMUM_SUPPORT = 0.1`, and then begin to mine the associated rules.

3.4.1 Rules by 'Minimum Support'

[36]:
```
# This defines what is considered as a minimum item
MINIMUM_SUPPORT = 0.1
frequent_items =␣
 ↪apriori(ARM_structured_data,min_support=MINIMUM_SUPPORT,use_colnames=True,␣
 ↪verbose=True)
display_markdown(frequent_items.head(5).append(frequent_items.tail(5)))
```

```
Processing 570 combinations | Sampling itemset size 5 4
```

	support	itemsets
0	0.495	frozenset({'has_bluetooth'})
1	0.5095	frozenset({'has_dual_sim'})
2	0.5215	frozenset({'has_four_g'})
3	0.7615	frozenset({'has_three_g'})
4	0.503	frozenset({'has_touch_screen'})
712	0.1165	frozenset({'has_three_g', 'has_four_g', 'screen_width_category_(-0.019, 4.5]', 'has_wifi'})

	support	itemsets
713	0.1165	frozenset({'has_three_g', 'ram_category_(3062.5, 3998.0]', 'has_four_g', 'is_expensive'})
714	0.1	frozenset({'has_three_g', 'clock_speed_category_(0.497, 1.125]', 'has_four_g', 'screen_width_category_(-0.019, 4.5]'})
715	0.11	frozenset({'has_three_g', 'front_cam_resolution_category_(0.981, 5.5]', 'has_four_g', 'px_height_category_(-1.9609999999999999, 490.0]'})
716	0.105	frozenset({'has_three_g', 'screen_width_category_(-0.019, 4.5]', 'has_four_g', 'px_height_category_(-1.9609999999999999, 490.0]'})

Comment: The (above) table demonstrates the item sets that meet the minimum defined support (`0.1`).

3.5 Confidence

In ARM, confidence is an indication of how often the rule has been found to be true. The confidence value of a rule, { $\boldsymbol{X}$ -> $\boldsymbol{Y}$ } with respect to a set of **z**'s {$\boldsymbol{Z}$}, is the proportion of the **z**'s that contains $\boldsymbol{X}$, which also contains $\boldsymbol{Y}$.

3.5.1 Rules by 'Minimum Confidence'

[37]:
```
ARM_COLS_OF_INTEREST = ['antecedents', 'consequents', 'support', 'confidence',␣
↪'lift']
```

[38]:
```
MINIMUM_CONFIDENCE_THRESHOLD = 0.20 # this is the minimum confidence threshold␣
↪to mine
rules_by_confidence = association_rules(frequent_items, metric="confidence",␣
↪min_threshold=MINIMUM_CONFIDENCE_THRESHOLD)

# Get all the rules that has mobile price
rules = rules_by_confidence[rules_by_confidence["consequents"].
↪map(set(['is_expensive']).issubset)].
↪sort_values("confidence",ascending=False)
display_markdown(rules[ARM_COLS_OF_INTEREST].reset_index(drop=True).head(10))
```

	antecedents	consequents	support	confidence	lift
0	frozenset({'ram_category_(3062.5, 3998.0]', 'has_four_g'})	frozenset({'is_expensive'})	0.1165	0.856618	3.42647
1	frozenset({'ram_category_(3062.5, 3998.0]', 'has_four_g'})	frozenset({'has_three_g', 'is_expensive'})	0.1165	0.856618	4.44996
2	frozenset({'has_three_g', 'ram_category_(3062.5, 3998.0]', 'has_four_g'})	frozenset({'is_expensive'})	0.1165	0.856618	3.42647
3	frozenset({'ram_category_(3062.5, 3998.0]', 'has_touch_screen'})	frozenset({'is_expensive'})	0.103	0.847737	3.39095

	antecedents	consequents	support	confidence	lift
4	frozenset({'has_three_g', 'ram_category_(3062.5, 3998.0]'})	frozenset({'is_expensive'})	0.1635	0.844961	3.37984
5	frozenset({'ram_category_(3062.5, 3998.0]', 'has_wifi'})	frozenset({'is_expensive'})	0.108	0.840467	3.36187
6	frozenset({'ram_category_(3062.5, 3998.0]', 'has_bluetooth'})	frozenset({'is_expensive'})	0.109	0.838462	3.35385
7	frozenset({'ram_category_(3062.5, 3998.0]'})	frozenset({'is_expensive'})	0.209	0.832669	3.33068
8	frozenset({'ram_category_(3062.5, 3998.0]', 'has_dual_sim'})	frozenset({'is_expensive'})	0.115	0.824373	3.29749
9	frozenset({'ram_category_(3062.5, 3998.0]'})	frozenset({'has_three_g', 'is_expensive'})	0.1635	0.651394	3.38387

Comment: The (above) table shows a min-confidence threshold of `0.65`, for the top `k=10` rules , and `0.43` for the top `k=16` rules (we have omitted 6, however see `association_rule_mining/rules_by_confidence_minimum_threshold_0.20.csv` for details.

3.5.2 Explained: `top-k` Rules

The `top-k` rules are the number (***k***) of rules (attribute patterns) that based on the confidence/support parameters, occur with the highest frequency in the dataset.

For example, according to the k=1 rule (sorted by `confidence`), if the phone has a `ram_category` that belongs in the range `3062.5 - 3998.0`, and ALSO has 4G (`has_four_g = True`), then there is a probability of 86% that this phone is expensive (`is_expensive = True`).

It should also be noted that the top 16 rules (`k=16`) all have an antecedent that contains `ram_category` in the `3062.5 - 3998.0` range.

[39]:
```
display_markdown(rules[ARM_COLS_OF_INTEREST].reset_index(drop=True).iloc[17:21])
```

	antecedents	consequents	support	confidence	lift
17	frozenset({'front_cam_resolution_category_(0.8, 5.5]'})	frozenset({'is_expensive'})	0.116	0.26484	1.05936
18	frozenset({'has_three_g', 'has_four_g'})	frozenset({'is_expensive'})	0.1375	0.263663	1.05465
19	frozenset({'has_four_g'})	frozenset({'has_three_g', 'is_expensive'})	0.1375	0.263663	1.36968
20	frozenset({'has_four_g'})	frozenset({'is_expensive'})	0.1375	0.263663	1.05465

Comment: The (above) table shows the 17th to the 20th rules, when the minimum confidence threshold is set to `0.26`. Here, we observe rules that do not reference `ram_category`, showing other attributes that can predict `is_expensive`.

3.6 Lift

Lift quantifies an association rule's ability to predict cases (as) possessing an improved response (against the population), measured against a random choice targeting model. An association rule is doing well (according to lift) if the response (within the target) exceeds the average for the population as a whole. Simply put, lift assesses the degree to which the occurrence of one (characteristic) "lifts" the occurrence of the other.

`lift (A, B) = P( A U B ) / P(A) P(B)`

`( lift < 1 )` → The occurrence of A is **negatively** correlated with the occurrence of B;

`( lift > 1 )` → The occurrence of A is **positively** correlated with the occurrence of B;

`( lift = 1 )` → The occurrence of A is **independent** of the occurrence of B;

3.6.1 Rules by 'Lift'

[41]:
```
MINIMUM_LIFT_THRESHOLD = -5

rules_by_lift = association_rules(frequent_items, metric="lift",␣
↪min_threshold=MINIMUM_LIFT_THRESHOLD)
rules_by_lift["translated_lift"] = rules_by_lift["lift"] - 1
rules_by_lift["is_translated_lift_negative"] = rules_by_lift["translated_lift"]␣
↪< 0
rules_by_lift["absolute_value_of_translated_lift"] = np.
↪abs(rules_by_lift["translated_lift"])

# Get all the rules that has mobile price
rules = rules_by_lift[rules_by_lift["consequents"].map(set(['is_expensive']).
↪issubset)].sort_values("absolute_value_of_translated_lift",ascending=False)
```

[43]:
```
display_markdown(rules[ARM_COLS_OF_INTEREST].reset_index(drop=True).head(10))
```

	antecedents	consequents	support	confidence	lift
0	frozenset({'ram_category_(3062.5, 3998.0]', 'has_four_g'})	frozenset({'has_three_g', 'is_expensive'})	0.1165	0.856618	4.44996
1	frozenset({'has_three_g', 'ram_category_(3062.5, 3998.0]'})	frozenset({'has_four_g', 'is_expensive'})	0.1165	0.602067	4.37867
2	frozenset({'ram_category_(3062.5, 3998.0]'})	frozenset({'is_expensive', 'has_dual_sim'})	0.115	0.458167	3.45787
3	frozenset({'ram_category_(3062.5, 3998.0]', 'has_four_g'})	frozenset({'is_expensive'})	0.1165	0.856618	3.42647
4	frozenset({'has_three_g', 'ram_category_(3062.5, 3998.0]', 'has_four_g'})	frozenset({'is_expensive'})	0.1165	0.856618	3.42647
5	frozenset({'ram_category_(3062.5, 3998.0]', 'has_touch_screen'})	frozenset({'is_expensive'})	0.103	0.847737	3.39095
6	frozenset({'ram_category_(3062.5, 3998.0]'})	frozenset({'has_three_g', 'is_expensive'})	0.1635	0.651394	3.38387

	antecedents	consequents	support	confidence	lift
7	frozenset({'has_three_g', 'ram_category_(3062.5, 3998.0]'})	frozenset({'is_expensive'})	0.1635	0.844961	3.37984
8	frozenset({'ram_category_(3062.5, 3998.0]'})	frozenset({'has_three_g', 'has_four_g', 'is_expensive'})	0.1165	0.464143	3.37559
9	frozenset({'ram_category_(3062.5, 3998.0]'})	frozenset({'has_four_g', 'is_expensive'})	0.1165	0.464143	3.37559

Comment: The (above) table shows top (`k=10`) rules, where we notice that `ram_category` is present in all (10) antecedents (see `association_rule_mining/rules_by_lift_sorted_by_absolute_value_of_translated_lift.csv` for the full list of rules).

We have ordered these rules by `absolute_value_of_translated_lift`, in order to better highlight the strongest attribute correlations (whether they be positive, or negative).

- The top `k=1` rule states that when a phone has `ram_category` in the range `3062.5 - 3998.0`, and also has 4g (`has_four_g=true`), then these attributes are highly positively correlated to the `has_three_g` and `is_expensive` attribute pair (lift=`4.449`).
- It should be noted that the top (`k=17`) rules all reference `ram_category` in the range `3062.5-3998.0`.

[44]:
```
display_markdown(rules[ARM_COLS_OF_INTEREST].reset_index(drop=True)[18:21])
```

	antecedents	consequents	support	confidence	lift
18	frozenset({'has_four_g'})	frozenset({'has_three_g', 'ram_category_(3062.5, 3998.0]', 'is_expensive'})	0.1165	0.223394	1.36632
19	frozenset({'has_three_g'})	frozenset({'has_four_g', 'is_expensive'})	0.1375	0.180565	1.3132
20	frozenset({'has_three_g'})	frozenset({'ram_category_(3062.5, 3998.0]', 'has_four_g', 'is_expensive'})	0.1165	0.152988	1.3132

Comment: The 18th up to the 20th (above) show different attribute with a lift that is positively correlated, but not as strong as the previous the rules.

3.7 Association Rule Mining - Recommendation for Designing an Expensive Phone

Based on our results from association rule mining, we can advise a manufacturer (who wishes to design an expensive phone) on the following features…

The most notable attribute in an item-set, that resulted in an expensive phone, contained RAM that fell in the RAM category (range) `3062.5- 3998.0`. However, within the same item-set/s as RAM, we noted the following (most frequent) attributes:

1. four G

2. touch scren
3. three G
4. wifi
5. bluetooth

…ranked by the strongest defining confidence. This discovery upholds the initial patterns that we observed in the graph of relative frequency of all attributes (in their ordinal form), divided by the `is_expensive` field.

4 Classification

Classification is the supervised-learning process of determining and assigning classes to data rows. This process can identify the class of an unknown row, based upon data row/s that a model has been trained upon.

Classification in `CITS 3401 - Project 2` will be done with cross validation of ten folds, using differing partitions of our dataset for testing and training, to avoid overfitting our model. Furthermore, the training will be done with a `RANDOM_STATE` seeding, for the express purpose of mitigating the uncontrollable inconsistency of algorithm that uses randomisation.

The classification will be done with both Decision Tree ('DT') and Support Vector Machine ('SVM'), with hyperparameter optimisation. The metric will be accuracy, as the attribute `is_expensive` is weighted equally (between the possible `True` or `False` values).

We will be using the `cleaned_data` dataset (not the wholly discretised dataset that was used in the Association Rule Mining), as SVM works well in seperating continuous attributes, whilst DT can take either continuous ('Regression Tree') or discrete ('Classification Tree') data.

[45]:
```
# Seperation of data into features and target
learning_data = cleaned_data.drop("id",axis="columns")
target_data = learning_data["is_expensive"]
feature_data = learning_data.drop("is_expensive",axis="columns")
```

4.1 Decision Tree

A decision tree is an algorithm that separates the data using different thresholds, within different attributes. There are two main criterium for choosing an attribute and a threshold:

- `Gini Index` -> Information Gain calculates effective change in entropy after making a decision based on the value of an attribute.
- `Information Gain (entropy)` -> The gini index, calculates the amount of probability of a specific feature that is classified incorrectly when selected randomly.

We will now configure and test the aforementioned classifiers…

[47]:
```
dt_pipeline_1 = Pipeline([
    ("dt_classifier",DecisionTreeClassifier(random_state=RANDOM_STATE))
])
# Grid Search
param_grid = {
```

```
    'dt_classifier__criterion': ["gini", "entropy"],
    'dt_classifier__max_depth': [None] + list(range(1,len(feature_data.
 ↪columns)))
}
```

[ ]:
```
dt_pipeline_1_search = GridSearchCV(dt_pipeline_1, param_grid, n_jobs=-1,␣
 ↪cv=CROSS_VALIDATION_PARTITION)
dt_pipeline_1_search.fit(feature_data,target_data)
```

[49]:
```
show_top_results(dt_pipeline_1_search)
```

	rank_test_score	mean_test_score	params
26	1	0.9475	{'dt_classifier__criterion': 'entropy', 'dt_classifier__max_depth': 6}
28	2	0.9435	{'dt_classifier__criterion': 'entropy', 'dt_classifier__max_depth': 8}
29	3	0.9415	{'dt_classifier__criterion': 'entropy', 'dt_classifier__max_depth': 9}
27	4	0.9405	{'dt_classifier__criterion': 'entropy', 'dt_classifier__max_depth': 7}
20	5	0.94	{'dt_classifier__criterion': 'entropy', 'dt_classifier__max_depth': None}

Comment: The (above) table shows the top 5 hyper-parameters of the decision tree, and the mean test scores of all CV folds. The best mean test score (`0.9475`) has Information Gain as a criterium, and a `decision_tree_max_depth = 6`. We see **no** test in the listed top 5 containing gini-index criterium.

4.1.1 Decision Tree Visualization

[51]:
```
viz = dtreeviz(dt_pipeline_1_search.best_estimator_[0], feature_data,␣
 ↪target_data,
                target_name="is_expensive",
                feature_names=feature_data.columns,
                class_names=list(target_data.unique()))

viz.save("decision_tree.svg");
#viz
```

Comment: The (above) diagram shows the decision tree visualisation. (See submission folder `diagrams/decision_tree.png`)

4.2 Support Vector Machine

The 'SVM' algorithm finds the best hyperplane in the multi-atribute dimension, to seperate data into multiple, relevant classes. The parameter/s available in SVM are:

- `Seperation Degree` of the Hyperplane (also known as `Kernel`);

[52]:
```
svm_pipeline_2 = Pipeline([
    ("svm_classifier",SVC())
])

# Grid Search
param_grid = {
    'svm_classifier__kernel': ['linear', 'poly', 'rbf', 'sigmoid']
}
```

[ ]:
```
svm_pipeline_2_search = GridSearchCV(svm_pipeline_2, param_grid, n_jobs=-1,␣
↪cv=CROSS_VALIDATION_PARTITION)
svm_pipeline_2_search.fit(feature_data,target_data)
```

[54]:
```
show_top_results(svm_pipeline_2_search)
```

	rank_test_score	mean_test_score	params
0	1	0.9895	{'svm_classifier___kernel': 'linear'}
1	2	0.9825	{'svm_classifier___kernel': 'poly'}
2	3	0.9785	{'svm_classifier___kernel': 'rbf'}
3	4	0.53	{'svm_classifier___kernel': 'sigmoid'}

Comment: The (above) table displays the top 5 hyperparameters, of the SVM. We observe the best mean test score of (`0.9895`) attributable to the `kernel = linear` hyperplane.

4.3 Comparison DT vs SVM

Our tests conclude that SVM yields results with higher accuracies, than DT. Though both (methods) are exposed to the same data, and their most valuable hyperparameters are optimized, we must acknowledge that SVM is a more appropriate algorithm for this dataset, and experiment.

5 Clustering

Clustering is the unsupervised-learning process of assigning data to a group, or 'cluster'. Clustering identifies similarities between objects, which it groups according to these characteristics in common, and which differentiate them from other groups of data. The clustering process is very similar to the classification process, aside from the classes not being known / labelled (in clustering).

5.1 Cluster of Size 2

The clustering method will begin with `CLUSTER_SIZE = 2`, enabling us to compare the two distinct values of `is_expensive`, and the relevant clusters. This **DOES NOT** mean that the `CLUSTER_SIZE` set at 2, is the optimal cluster number.

Example: *There may be a phone that sits in between* `is_expensive=True` *and* `is_expensive=False` *(a medium priced phone) that a cluster size* `n=2` *would not be able to properly represent.*

5.2 Kmeans

`K-means` clustering aims to partition observations (data rows) into k clusters, in which each observation belongs to the cluster with the nearest cluster-mean / cluster-centroid.

Confusion Table

[57]:
```
kmeans_model = KMeans(n_clusters=2, random_state=RANDOM_STATE)
kmeans_model.fit(feature_data)
kmeans_prediction = kmeans_model.labels_
```

[59]:
```
print_confusion_table(crosstab, accuracy)
```

```
prediction    0    1
target
False       977  523
True          0  500
--------------------------
Accuracy: 0.7385 (73.85%)
--------------------------
```

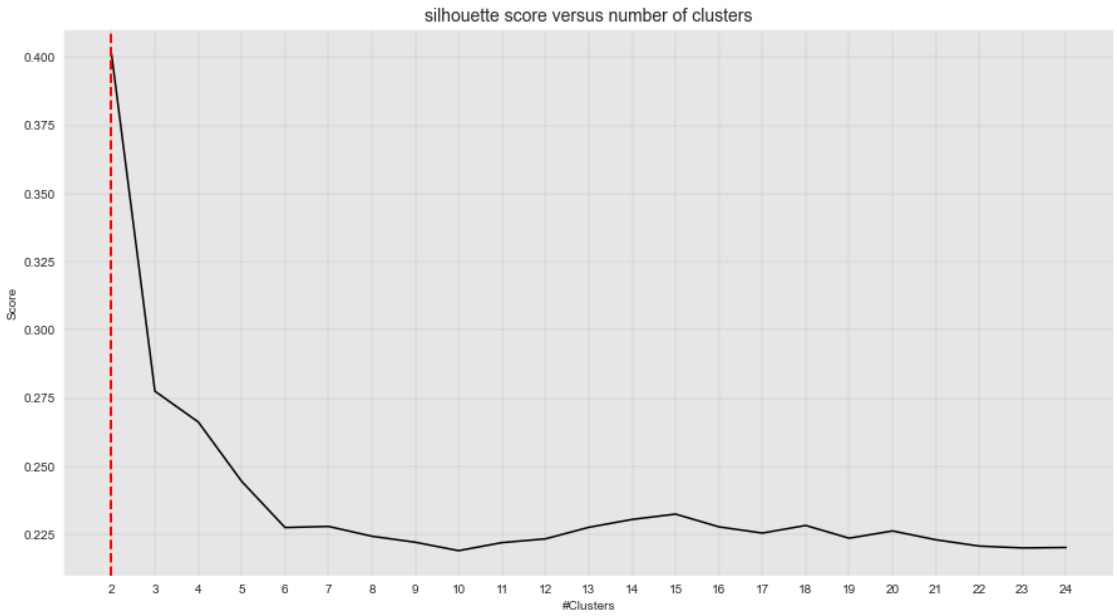
Comment: The confusion table (above) shows a `73.85%` accuracy when matched with the labelled groups, wherein the other 26.15% inaccuracy can be explained by kmeans predicting a phone `is_expensive` when it is in fact not (false positive), and 0% for false negative.

5.3 Validation of Two Cluster Size

In the previous heading, it is assumed that the cluster of size 2 is expected. This is an extra exploratory step to validate whether a cluster of size 2 is the best way to split the data.

[60]:
```
cluster_evaluation = clusteval(method='silhouette',cluster="kmeans")
cluster_evaluation.fit(np.array(feature_data))
cluster_evaluation.plot()
```

```
  0%|
| 0/23 [00:00<?, ?it/s]


[clusteval] >Fit using kmeans with metric: euclidean, and linkage: ward
[clusteval] >Evaluate using silhouette.

100%|
     | 23/23 [00:06<00:00,  3.29it/s]

[clusteval] >Optimal number clusters detected: [2].
[clusteval] >Fin.
```



[60]:
```
(<Figure size 1080x576 with 1 Axes>,
 <AxesSubplot:title={'center':'silhouette score versus number of clusters'},
xlabel='#Clusters', ylabel='Score'>)
```

5.3.1 Silhouette Score

Silhouette score refers to a method of interpretation and validation of consistency within clusters of data. The higher the score, the better the overall cluster measure, such as within-variability and

between-variability of the clusters. According to the chart (above) a `CLUSTER_SIZE = 2` is most appropriate.

6 Data Reduction

More data is not always the solution. When working with 'big data' we must consider the issues around redundancy, outliers, and storage. Data reduction (as the name suggests) actively encourages the reduction of data, where possible, through the elimination of data-rows (numerosity), and data-columns (features). This reduction must be weighed in against the overall quality/information that would be forfeited.

6.1 Reasons for Reducing Data

- Increases storage capacity
- Easy and efficient Mining, reduces time and memory requirement
- Easy visualisation
- Help to eliminate irrelevant /redundant features
- Reduces noise

6.2 Numerosity Reduction

Numerosity reduction involves the replacement of voluminous data, with an alternate, smaller form of data representation. This exchange can be achieved via parametric and non-parametric methods.

- `Parametric Numerosity Reduction` -> These techniques include linear regression and log linear models, to which the model's parameters can be stored, instead of the FULL data representation.
- `Non-Parametric Numerosity Reduction` -> Sampling, histograms, clustering, data cube aggregation.

6.2.1 Numerosity Reduction - Sampling with Decision Tree

[62]:
```
sampled_learning_data = learning_data.sample(frac=0.5,random_state=RANDOM_STATE)
sampled_target_data = sampled_learning_data["is_expensive"]
sampled_feature_data = sampled_learning_data.drop("is_expensive",axis="columns")
```

[ ]:
```
dt_pipeline_1_search = GridSearchCV(dt_pipeline_1, param_grid, n_jobs=-1,␣
↪cv=CROSS_VALIDATION_PARTITION)
dt_pipeline_1_search.fit(sampled_feature_data,sampled_target_data)
```

[64]:
```
# Results in Tabular format
show_top_results(dt_pipeline_1_search)
```

	rank_test_score	mean_test_score	params
0	1	0.928	{'dt_classifier___criterion': 'gini', 'dt_classifier___max_depth': None}
10	1	0.928	{'dt_classifier___criterion': 'gini', 'dt_classifier___max_depth': 10}

	rank_test_score	mean_test_score	params
19	1	0.928	{'dt_classifier___criterion': 'gini', 'dt_classifier___max_depth': 19}
18	1	0.928	{'dt_classifier___criterion': 'gini', 'dt_classifier___max_depth': 18}
17	1	0.928	{'dt_classifier___criterion': 'gini', 'dt_classifier___max_depth': 17}

Comment: Though we see (above) a drop in `mean_test_score` by `1.2%` (which is not surprising), the amount of data needed for testing and training has been cut by `50%`.

6.2.2 Numerosity Reduction - Sampling with Support Vector Machine

[65]:
```
svm_pipeline_2 = Pipeline([
    ("svm_classifier",SVC())
])

# Grid Search
param_grid = {
    'svm_classifier__kernel': ['linear', 'poly', 'rbf', 'sigmoid']
}
```

[ ]:
```
svm_pipeline_2_search = GridSearchCV(svm_pipeline_2, param_grid, n_jobs=-1,␣
↪cv=CROSS_VALIDATION_PARTITION)
svm_pipeline_2_search.fit(sampled_feature_data,sampled_target_data)
```

[67]:
```
# Results in Tabular format
show_top_results(svm_pipeline_2_search)
```

	rank_test_score	mean_test_score	params
0	1	0.982	{'svm_classifier___kernel': 'linear'}
1	2	0.977	{'svm_classifier___kernel': 'poly'}
2	3	0.974	{'svm_classifier___kernel': 'rbf'}
3	4	0.548	{'svm_classifier___kernel': 'sigmoid'}

Comment: (Above) Similar to the results from the numerosity reduction with the Decision Tree, we have observed an insignificant drop in accuracy from the default SVM model (accuracy of `98.95%`) to the numerosity reduced SVM model (accuracy `98.2%`).

6.3 Attribute Reduction & Attribute Selection

Attribute reduction focuses on (no surprise) the reduction of the data's columns/attributes. Data in a high-dimensional space will be transformed (by attribute reduction methods) into a low-dimensional space, so that this new, low-dimensional representation may retain appropriate amounts of meaningful properties (from the original data), whilst **also** heavily reducing the

computational-power-requirements (storage, navigation, transformation etc).

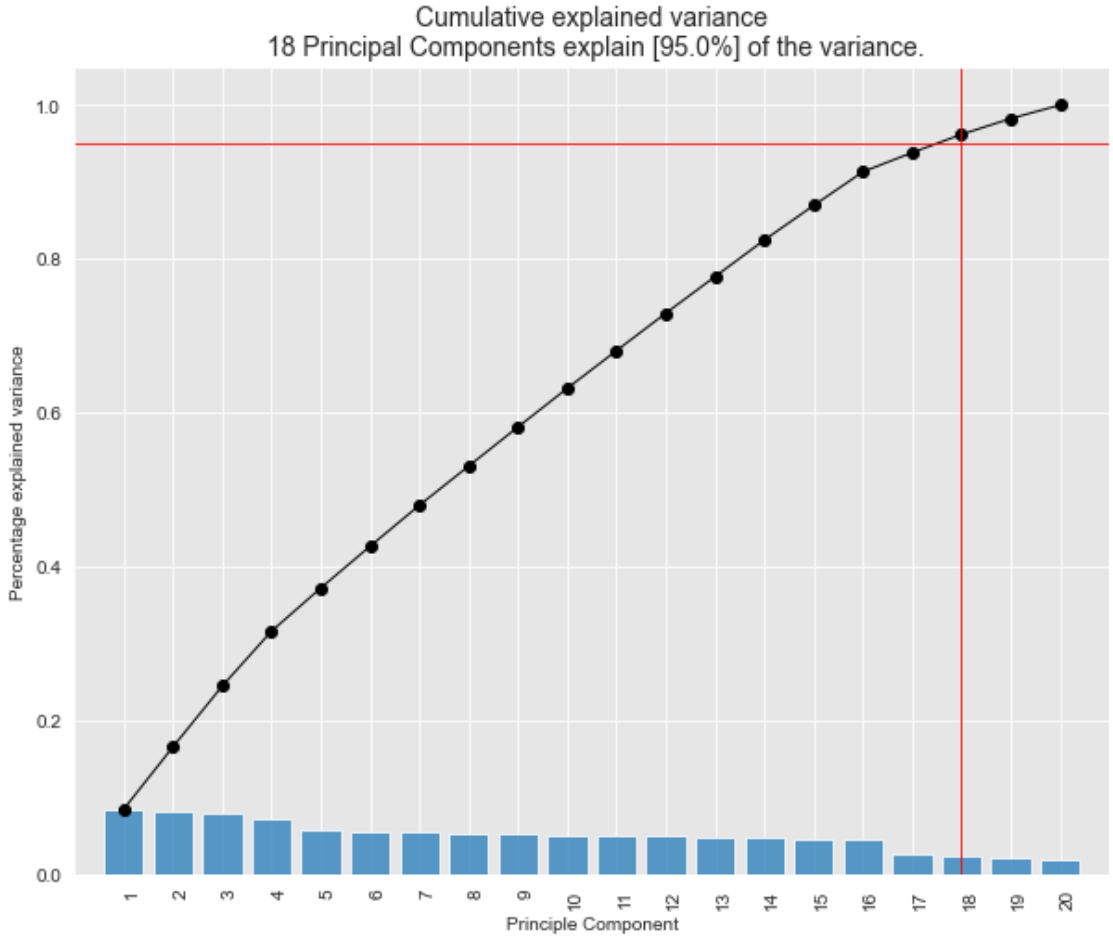
It achieves this mainly through discovering the attributes that possess the lowest 'predictive value', i.e. they contribute little to the domain specific question or are insignificant in aiding attributes that *are* contributing.

6.3.1 Principal Component Analysis (PCA)

Variance -> In the field of statistics, variance is one of the most important measures, as it examines how the data varies within (internally) and inbetween (interactively) attributes.

PCA summarises the attributes of the model through linear combinations expressed as 'principal components' ('PC') that maximises the variance. The variance contribution of the principal components (to the data) are ordered such that the first principal component is the 'most valuable' or 'highest contributory' attribute. Furthermore, one of the requirements of PCA is that the data is scaled, as the linear combination is sensitive to (attribute value) ranges (values with large magnitude value ranges will dominate the other PCs).

It should be noted that although PCA does a good job of multidimensional to low-dimensional data summarisation, it makes it ***very*** difficult to determine which domain specific attribute/s is the one/s contributing to the model.



Comment: According to the total contributions, to be able to explain 95% of the data's variance, there is a requirement to retain 18 principal components. It should be noted that this is most unusual for PCA, with most PCA's having 95% of the (data's) variance explained by only a tiny subset of the data's total dimensions.

6.3.2 Attribute Reduction - Decision Tree with PCA

[ ]:
```
dt_pipeline_3 = Pipeline([
    ("scale", StandardScaler()),
    ("pca", PCA()),
    ("dt_classifier",DecisionTreeClassifier(random_state=RANDOM_STATE))
])
# Grid Search
param_grid = {
    'pca__n_components': range(1,len(feature_data.columns)),
    'dt_classifier__criterion': ["gini", "entropy"],
    'dt_classifier__max_depth': [None] + list(range(1,len(feature_data.
 ↪columns)))
}
```

[ ]:
```
dt_pipeline_3_search = GridSearchCV(dt_pipeline_3, param_grid, n_jobs=-1,␣
 ↪cv=CROSS_VALIDATION_PARTITION)
dt_pipeline_3_search.fit(feature_data,target_data)
```

[71]:
```
show_top_results(dt_pipeline_3_search)
```

	rank_test_score	mean_test_score	params
568	1	0.8395	{'dt_classifier__criterion': 'entropy', 'dt_classifier__max_depth': 9, 'pca__n_components': 18}
682	2	0.8385	{'dt_classifier__criterion': 'entropy', 'dt_classifier__max_depth': 15, 'pca__n_components': 18}
567	3	0.8385	{'dt_classifier__criterion': 'entropy', 'dt_classifier__max_depth': 9, 'pca__n_components': 17}
566	3	0.8385	{'dt_classifier__criterion': 'entropy', 'dt_classifier__max_depth': 9, 'pca__n_components': 16}
663	5	0.838	{'dt_classifier__criterion': 'entropy', 'dt_classifier__max_depth': 14, 'pca__n_components': 18}

Comment: The results from our Decision Tree attribute reduction with PCA demonstrated a **significant** decrease in accuracy (from un-reduced accuracy of `94.75%`, to `83.95%`. These results come as no surprise, as per results earlier, there is an unusual contribution to total variance of PCA. We should also consider the possibility that the summarisation of attributes to principal components increased the impact that outliers exerted upon the (outlier sensitive) decision tree model.

6.3.3 Attribute Reduction - Support Vector Machine with PCA

[ ]:
```
svm_pipeline_3 = Pipeline([
    ("scale", StandardScaler()),
    ("pca", PCA()),
    ("svm_classifier",SVC())
])

# Grid Search
param_grid = {
    'pca__n_components': range(1,len(feature_data.columns)),
    'svm_classifier__kernel': ['linear', 'poly', 'rbf', 'sigmoid',␣
 ↪'precomputed']
}
```

[ ]:
```
svm_pipeline_3_search = GridSearchCV(svm_pipeline_3, param_grid, n_jobs=-1,␣
 ↪cv=CROSS_VALIDATION_PARTITION)
svm_pipeline_3_search.fit(feature_data,target_data)
```

[75]:
```
show_top_results(svm_pipeline_3_search)
```

	rank_test_score	mean_test_score	params
85	1	0.991	{'pca___n_components': 18, 'svm_classifier___kernel': 'linear'}
80	1	0.991	{'pca___n_components': 17, 'svm_classifier___kernel': 'linear'}
90	3	0.9895	{'pca___n_components': 19, 'svm_classifier___kernel': 'linear'}
75	4	0.984	{'pca___n_components': 16, 'svm_classifier___kernel': 'linear'}
88	5	0.9785	{'pca___n_components': 18, 'svm_classifier___kernel': 'sigmoid'}

Comment: The results (above) show the minimal improvement in accuracy, when the SVM model undertook the attribute reduction process. SVM is more robust to noise and outliers (than Decision Trees), and so it may be possible that contrary to above, it increased the impact of the "good quality" data, rather than the outlier data.

Prior to reduction, we observed an accuracy of `98.95%`, which then increased to `99.1%`.

6.4 Select K-Best with Information Gain (Entropy) and ANOVA F-statistic

There are multiple ways to select the best attribute/s of a dataset, mainly through the removal of redundant (highly-corellated) or irrelevant features. The common methods include 'F-statistic', and 'Information Gain'. F-statistic iteratively picks the attribute that *maximises* the varaiance (rather than a combination of attributes, seen in PCA).

Information gain measures the entropy-information available in a probability distribution. E.g :

- Skewed Probability Distribution (unsurprising) -> Low entropy.
- Balanced Probability Distribution (surprising) -> High entropy.

6.4.1 Attribute Reduction - Decision Tree with Select K-Best

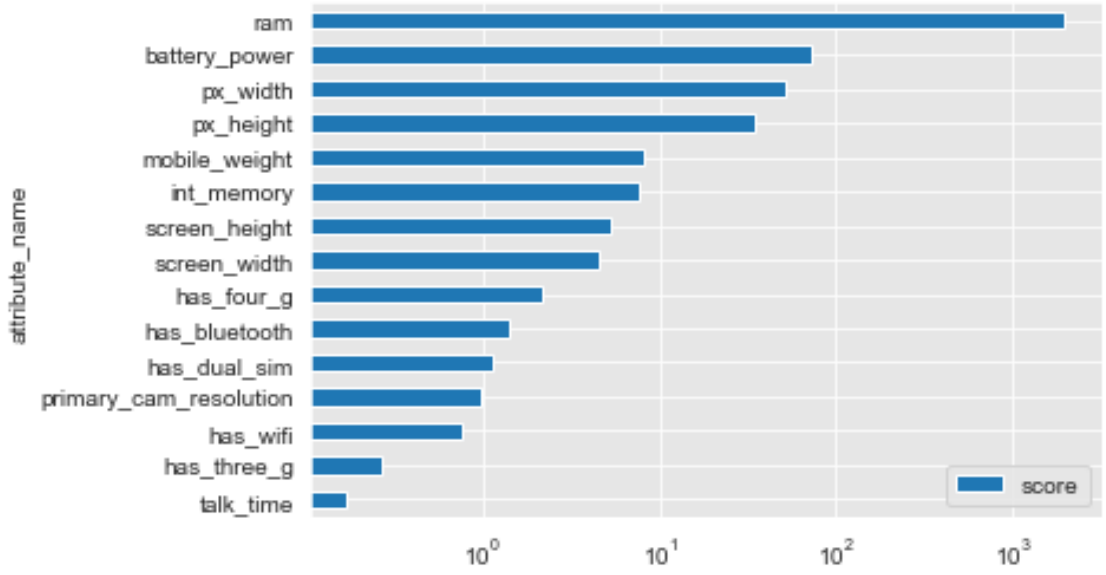
[77]:
```
dt_pipeline_4 = Pipeline([
    ("select_kbest", SelectKBest()),
    ("dt_classifier",DecisionTreeClassifier(random_state=RANDOM_STATE))
])
# Grid Search
param_grid = {
    'select_kbest__k': range(1,len(feature_data.columns)),
    'select_kbest__score_func': [f_classif,mutual_info_classif], # F-value␣
 ↪statistic and Information Gain (entropy)
    'dt_classifier__criterion': ["entropy"], # - Entropy is better than GINI␣
 ↪from previous results
    'dt_classifier__max_depth': [None] + list(range(1,len(feature_data.
 ↪columns)))
}
```

[ ]:
```
dt_pipeline_4_search = GridSearchCV(dt_pipeline_4, param_grid, n_jobs=-1,␣
 ↪cv=CROSS_VALIDATION_PARTITION)
dt_pipeline_4_search.fit(feature_data,target_data)
```

[79]:
```
show_top_results(dt_pipeline_4_search)
```

	rank_test_score	mean_test_score	params
332	1	0.9515	{'dt_classifier___criterion': 'entropy', 'dt_classifier___max_depth': 8, 'select_kbest___k': 15, 'select_kbest___score_func': <function f_classif at 0x00000207300F1AF0>}
250	2	0.9515	{'dt_classifier___criterion': 'entropy', 'dt_classifier___max_depth': 6, 'select_kbest___k': 12, 'select_kbest___score_func': <function f_classif at 0x00000207300F1AF0>}
260	2	0.9515	{'dt_classifier___criterion': 'entropy', 'dt_classifier___max_depth': 6, 'select_kbest___k': 17, 'select_kbest___score_func': <function f_classif at 0x00000207300F1AF0>}
256	2	0.9515	{'dt_classifier___criterion': 'entropy', 'dt_classifier___max_depth': 6, 'select_kbest___k': 15, 'select_kbest___score_func': <function f_classif at 0x00000207300F1AF0>}
248	5	0.951	{'dt_classifier___criterion': 'entropy', 'dt_classifier___max_depth': 6, 'select_kbest___k': 11, 'select_kbest___score_func': <function f_classif at 0x00000207300F1AF0>}

6.4.2 Attribute Selection with F-Statistic prior to Decision Tree



Comments: The table and graph (above) display the small improvement in the accuracy percentge, when the Decision Tree model experienced the 'select k-best' attribute reduction (an increase from `94.75%` to `95.15%`).

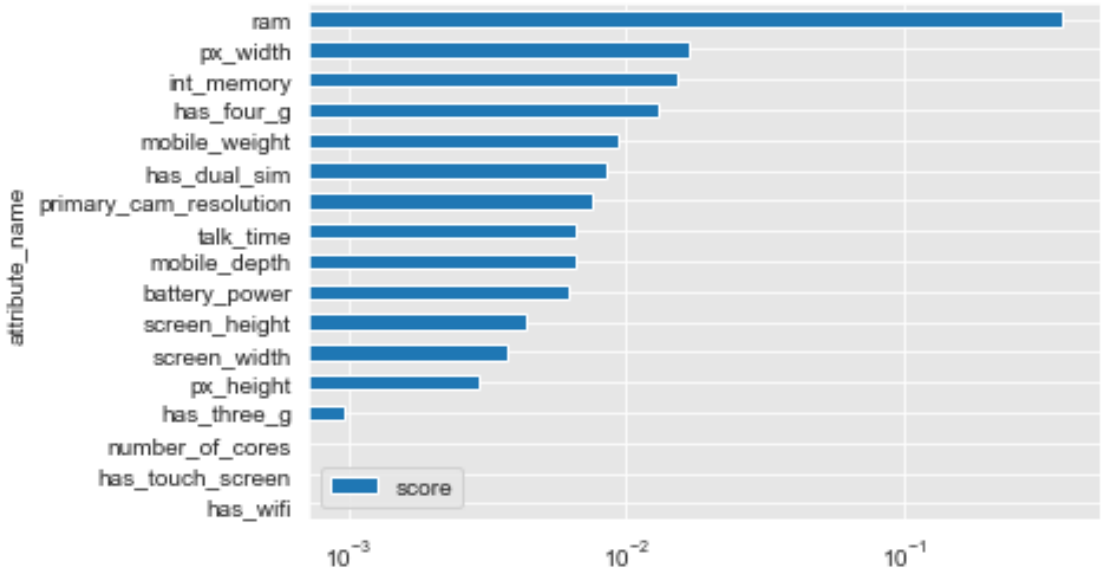
We see the reduction method has left us with 15 attributes, in comparison to the 18 attributes in PCA.

In this scenario, it appears that F-statistic is 'better' than information gain, with regard to the transformation. Note however, that information gain is later used for the decision tree classifier.

6.4.3 Attribute Selection - With Information Gain, Prior to Decision Tree

[84]:
```
select_k_best = dt_pipeline_4_search.best_estimator_[0]
attributes_df = pd.DataFrame({"attribute_name": feature_data.
 ↪columns[select_k_best.get_support()],"score": select_k_best.
 ↪scores_[select_k_best.get_support()]})
attributes_df.sort_values("score").
 ↪plot(x="attribute_name",y="score",kind="barh", logx=True)
print(f"Number of Attributes: {len(attributes_df)}")
```

```
Number of Attributes: 17
```



Comment: The analysis above shows that F-statistic proved to be the method that makes the accuracy of the model higher (94.90%). This is an improvement, but not as 'good' as the improvement experienced with F-statistic. However, for exploratory, examining the attributes there are differences in the attributes prioritised by information gain and F-statistic.

6.4.4 Attribute Reduction - Support Vector Machine with Select K-Best

[85]:
```
svm_pipeline_4 = Pipeline([
    ("select_kbest", SelectKBest()),
    ("svm_classifier",SVC())
])

# Grid Search
param_grid = {
    'select_kbest__k': range(10, 18),
    'select_kbest__score_func': [f_classif,mutual_info_classif], # F-value␣
 ↪statistic and Information Gain (entropy)
    'svm_classifier__kernel': ['linear', 'rbf'] # removed other attributes that␣
 ↪seems to be inaccurate
}
```

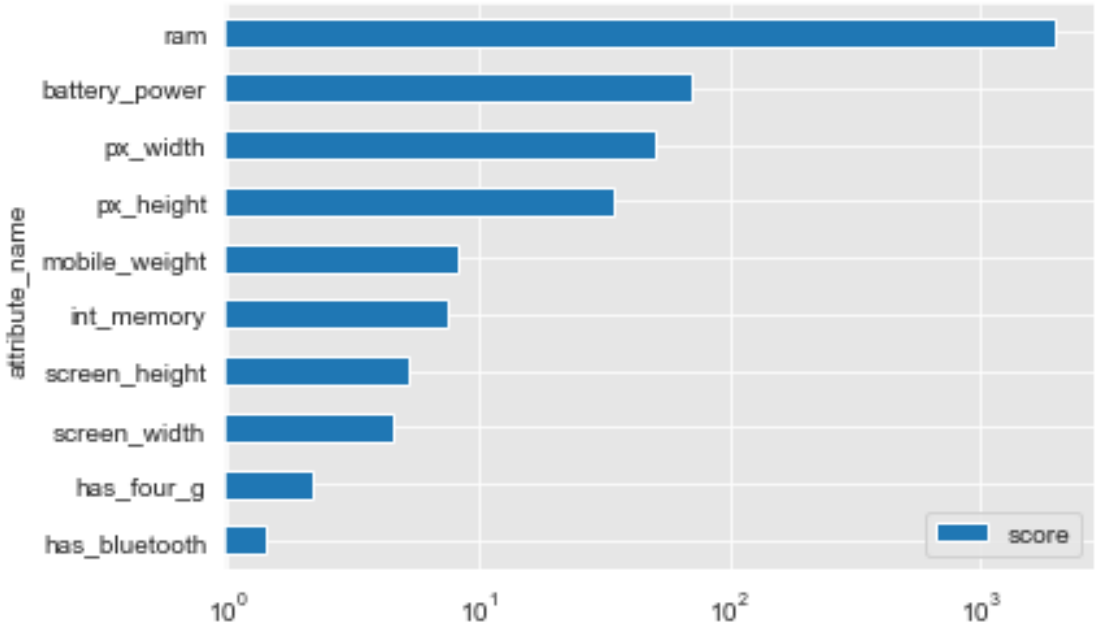
[ ]:
```
svm_pipeline_4_search = GridSearchCV(svm_pipeline_4, param_grid, n_jobs=-1,␣
 ↪cv=CROSS_VALIDATION_PARTITION)
svm_pipeline_4_search.fit(feature_data,target_data)
```

[87]:
```
show_top_results(svm_pipeline_4_search)
```

	rank_test_score	mean_test_score	params
0	1	0.9915	{'select__kbest___k': 10, 'select__kbest___score_func': <function f_classif at 0x00000207300F1AF0>, 'svm__classifier___kernel': 'linear'}
4	1	0.9915	{'select__kbest___k': 11, 'select__kbest___score_func': <function f_classif at 0x00000207300F1AF0>, 'svm__classifier___kernel': 'linear'}
12	3	0.991	{'select__kbest___k': 13, 'select__kbest___score_func': <function f_classif at 0x00000207300F1AF0>, 'svm__classifier___kernel': 'linear'}
8	4	0.9905	{'select__kbest___k': 12, 'select__kbest___score_func': <function f_classif at 0x00000207300F1AF0>, 'svm__classifier___kernel': 'linear'}
16	4	0.9905	{'select__kbest___k': 14, 'select__kbest___score_func': <function f_classif at 0x00000207300F1AF0>, 'svm__classifier___kernel': 'linear'}

[89]:
```
select_k_best = svm_pipeline_4_search.best_estimator_[0]
attributes_df = pd.DataFrame({"attribute_name": feature_data.
 ↪columns[select_k_best.get_support()],"score": select_k_best.
 ↪scores_[select_k_best.get_support()]})
attributes_df.sort_values("score").
 ↪plot(x="attribute_name",y="score",kind="barh", logx=True)
print(f"Number of Attributes: {len(attributes_df)}")
```

```
Number of Attributes: 10
```



Comment: The output (above) indicates a minor accuracy improvement, from `98.95%` to `99.1%`. Note that this increase is `0.04%` better than the observed accuracy for SVM, with PCA.

We also observe that the suggested best number of attributes (for SVM) from F-statistic is `10`, a notably smaller value than the suggested 'best number' of attributes from F-statistic for Decision Tree (`15`).

6.5 Data Reduction - Conclusion

As seen in our experiments above (with our chosen measure of *classifier-accuracy* as accuracy), numerosity reduction **heavily** decreases the required data, in exchange for a small decrease in overall model accuracy. Attribute reduction via transformation (such as PCA) is seen in varying instances to both increase / decrease the model accuracy, while attribute reduction (by selection) demonstrated an overall net increase in accuracy . Furthermore, with attribute reduction, we also observed a decrease in required model-training time.

6.6 Attribute Selection - Manual Selection

With user-side specific domain knowledge (regarding expensive phones), we manually selected the following attributes. Note that the accuracy of this "model" is heavily dependent on the depth of the user's domain specific knowledge. In most real-world data science projects, manual attribute selection (performed by subject matter experts), mitigates the possibility of over-fitting, that arises with pure criterium-based attribute selection.

For example, in electrical-engineering problems (where the voltage attribute is crucial) the data may suggest that voltage could/should be eliminated (for the purpose of increasing accuracy). If relied upon, the resultant model has the 'best' accuracy, but it may inappropriate and/or unusable in the real world.

[90]:
```
# Based on domain-specific knowledge and the exploratory plots
MANUALLY_SELECTED_ATTRIBUTES = [
    "ram",
    "has_four_g",
    "mobile_depth",
    "screen_width",
    "screen_height",
    "primary_cam_resolution",
    "number_of_cores",
    "int_memory",
    "px_width",
    "px_height"]

feature_data_manually_selected = learning_data.
 ↪drop("is_expensive",axis="columns")[MANUALLY_SELECTED_ATTRIBUTES]
```

6.7 Decision Tree with Manually Selected Attribute

[ ]:
```
dt_pipeline_1_search = GridSearchCV(dt_pipeline_1, param_grid, n_jobs=-1,␣
↪cv=CROSS_VALIDATION_PARTITION)
dt_pipeline_1_search.fit(feature_data_manually_selected,target_data)
```

[93]:
```
show_top_results(dt_pipeline_1_search)
```

	rank_test_score	mean_test_score	params
4	1	0.9235	{'dt_classifier___criterion': 'gini', 'dt_classifier___max_depth': 4}
2	2	0.9215	{'dt_classifier___criterion': 'gini', 'dt_classifier___max_depth': 2}
1	2	0.9215	{'dt_classifier___criterion': 'gini', 'dt_classifier___max_depth': 1}
3	4	0.9205	{'dt_classifier___criterion': 'gini', 'dt_classifier___max_depth': 3}
25	5	0.9175	{'dt_classifier___criterion': 'entropy', 'dt_classifier___max_depth': 5}

Comment: When the Decision Tree was reduced with our manually-selected-attribute model, we note a relatively large drop in accuracy from `94.75%` to `92.35%` (`2.4%`).

6.8 SVM with Manually Selected Attributes

[ ]:
```
svm_pipeline_2_search = GridSearchCV(svm_pipeline_2, param_grid, n_jobs=-1,␣
↪cv=CROSS_VALIDATION_PARTITION)
svm_pipeline_2_search.fit(feature_data_manually_selected,target_data)
```

[97]:
```
show_top_results(svm_pipeline_2_search)
```

	rank_test_score	mean_test_score	params
2	1	0.9295	{'svm_classifier___kernel': 'rbf'}
0	2	0.9265	{'svm_classifier___kernel': 'linear'}
1	2	0.9265	{'svm_classifier___kernel': 'poly'}
3	4	0.519	{'svm_classifier___kernel': 'sigmoid'}

Comment: When the SVM was reduced with our manually-selected-attribute model, we observe a notably large drop in accuracy from `98.95%` to `92.95%`. This decrease in accuracy with SVM (`6.0%`) is much larger than the decrease experienced by Decision Tree.

6.9 Attribute Selection - Conclusion

Through our experiments, F-statistic attribute selection demonstrated the best overall accuracy for both SVM and Decision Tree, as defined by the hyperparameter grid search.

Our results would indicate that Information Gain ranks second in accuracy, increasing both SVM and DT's accuracy. This can be contrasted with manual attribute selection, which showed poor results, with decreases in the accuracy of **both** of our classifiers.

7 Appendix

Raw Data Profiling Report

[ ]:
```
raw_profile = ProfileReport(raw_data, explorative=True, orange_mode=True,␣
↪title="Raw Data Profiling Report")

# set the Metadata
metadata_dict = raw_metadata.to_dict()["Explaination"]

raw_profile.set_variable("variables.descriptions",metadata_dict)
raw_profile.to_file("./profile_reports/raw_data_profile.html")
# raw_profile
```

Clean Data Profiling Report

[ ]:
```
# Generate new pandas-profiling
cleaned_data_profile = ProfileReport(cleaned_data, explorative=True,␣
↪orange_mode=True, title="Clean Data Profiling Report")
cleaned_data_profile.set_variable("variables.
↪descriptions",cleaned_metadata_dict) # Set Metadata
cleaned_data_profile.to_file("./profile_reports/cleaned_data_profile.html")
# cleaned_data_profile
```

Discretised Data Profiling Report

[ ]:
```
# Generate new pandas-profiling
discretised_data_profile = ProfileReport(discretised_data, explorative=True,␣
↪orange_mode=True, title="Discretised Data Profiling Report")
discretised_data_profile.set_variable("variables.
↪descriptions",cleaned_metadata_dict) # Set Metadata
discretised_data_profile.to_file("./profile_reports/discretised_data_profile.
↪html")
# discretised_data_profile
```

Overview

Overview | Variables | Warnings 5 | Reproduction

Dataset statistics

Number of variables	22
Number of observations	2000
Missing cells	0
Missing cells (%)	0.0%
Duplicate rows	0
Duplicate rows (%)	0.0%
Total size in memory	747.1 KiB
Average record size in memory	382.5 B

Variable types

Numeric	15
Categorical	7

Variables

id

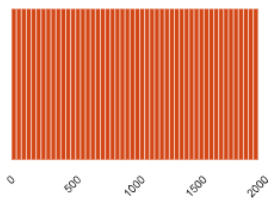
Real number ($\mathbb{R}_{\geq 0}$)

`UNIFORM`
`UNIQUE`

ID

Distinct	2000
Distinct (%)	100.0%
Missing	0
Missing (%)	0.0%
Infinite	0
Infinite (%)	0.0%

Mean	999.5
Minimum	0
Maximum	1999
Zeros	1
Zeros (%)	< 0.1%
Memory size	15.8 KiB



Toggle details

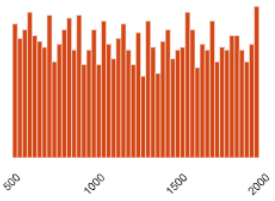
battery_power

Real number ($\mathbb{R}_{\geq 0}$)

Total energy a battery can store in one time measured in mAh

Distinct	1094
Distinct (%)	54.7%
Missing	0
Missing (%)	0.0%
Infinite	0
Infinite (%)	0.0%

Mean	1238.5185
Minimum	501
Maximum	1998
Zeros	0
Zeros (%)	0.0%
Memory size	15.8 KiB



Toggle details

Statistics | Histogram | Common values | Extreme values

Quantile statistics

Descriptive statistics

Minimum	501
5-th percentile	570.95
Q1	851.75
median	1226
Q3	1615.25
95-th percentile	1930.15
Maximum	1998
Range	1497
Interquartile range (IQR)	763.5

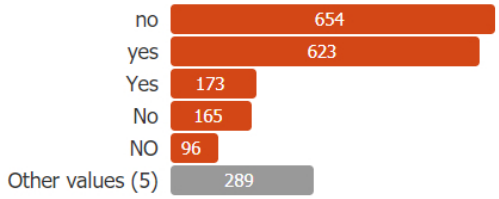
Standard deviation	439.4182061
Coefficient of variation (CV)	0.3547934133
Kurtosis	-1.224143883
Mean	1238.5185
Median Absolute Deviation (MAD)	382
Skewness	0.03189847179
Sum	2477037
Variance	193088.3598
Monotocity	Not monotonic

blue

Categorical

Has bluetooth or not

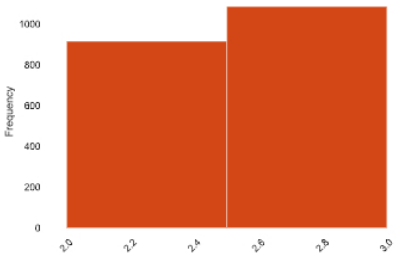
Distinct	10
Distinct (%)	0.5%
Missing	0
Missing (%)	0.0%
Memory size	116.4 KiB



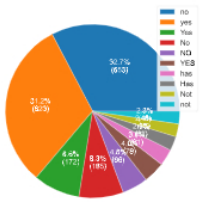
Toggle details

Overview | Categories | Words | Characters

Value	Count	Frequency (%)
no	654	32.7%
yes	623	31.1%
Yes	173	8.6%
No	165	8.2%
NO	96	4.8%
YES	79	4.0%
has	62	3.1%
Has	53	2.6%
Not	48	2.4%
not	47	2.4%



Histogram of lengths of the category



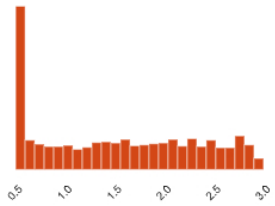
clock_speed

Real number ($R_{>0}$)

speed at which microprocessor executes instructions

Distinct	26
Distinct (%)	1.3%
Missing	0
Missing (%)	0.0%
Infinite	0
Infinite (%)	0.0%

Mean	1.52225
Minimum	0.5
Maximum	3
Zeros	0
Zeros (%)	0.0%
Memory size	15.8 KiB



Toggle details

Statistics | Histogram | Common values | Extreme values

Quantile statistics

Minimum	0.5
5-th percentile	0.5
Q1	0.7
median	1.5
Q3	2.2
95-th percentile	2.8
Maximum	3
Range	2.5
Interquartile range (IQR)	1.5

Descriptive statistics

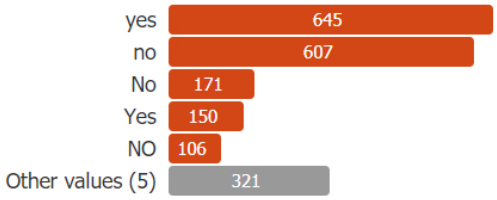
Standard deviation	0.8160042089
Coefficient of variation (CV)	0.5360513772
Kurtosis	-1.323417222
Mean	1.52225
Median Absolute Deviation (MAD)	0.8
Skewness	0.1780841203
Sum	3044.5
Variance	0.6658628689
Monotocity	Not monotonic

dual_sim

Categorical

Has dual sim support or not

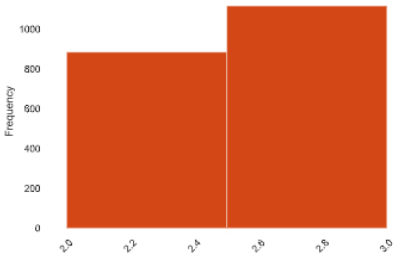
Distinct	10
Distinct (%)	0.5%
Missing	0
Missing (%)	0.0%
Memory size	116.4 KiB



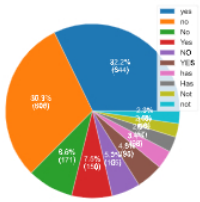
Toggle details

Overview | Categories | Words | Characters

Value	Count	Frequency (%)
yes	645	32.2%
no	607	30.3%
No	171	8.6%
Yes	150	7.5%
NO	106	5.3%
YES	99	5.0%
has	68	3.4%
Has	57	2.9%
Not	51	2.5%
not	46	2.3%



Histogram of lengths of the category



fc

Real number ($R_{\geq 0}$)

ZEROS

Distinct	20
Distinct (%)	1.0%
Missing	0

Mean	4.3095
Minimum	0
Maximum	19

Front Camera mega pixels

Missing (%)	0.0%
Infinite	0
Infinite (%)	0.0%

Zeros	474
Zeros (%)	23.7%
Memory size	15.8 KiB

Toggle details

Statistics | Histogram | Common values | Extreme values

Quantile statistics

Minimum	0
5-th percentile	0
Q1	1
median	3
Q3	7
95-th percentile	13
Maximum	19
Range	19
Interquartile range (IQR)	6

Descriptive statistics

Standard deviation	4.341443748
Coefficient of variation (CV)	1.007412402
Kurtosis	0.2770763246
Mean	4.3095
Median Absolute Deviation (MAD)	3
Skewness	1.019811411
Sum	8619
Variance	18.84813382
Monotocity	Not monotonic

four_g

Categorical

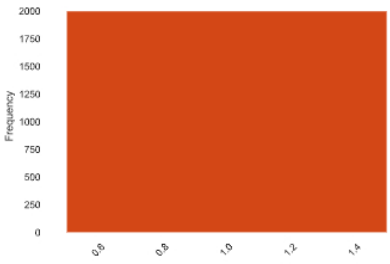
Has 4G or not. 1 = yes , 0 = no

Distinct	2
Distinct (%)	0.1%
Missing	0
Missing (%)	0.0%
Memory size	113.4 KiB

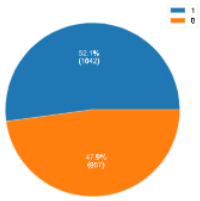
Toggle details

Overview | Categories | Words | Characters

Value	Count	Frequency (%)
1	1043	52.1%
0	957	47.9%



Histogram of lengths of the category



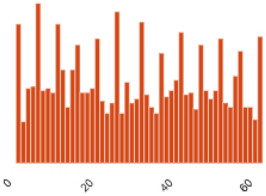
int_memory

Real number ($R_{\geq 0}$)

internal Memory in Gigabytes

Distinct	63
Distinct (%)	3.1%
Missing	0
Missing (%)	0.0%
Infinite	0
Infinite (%)	0.0%

Mean	32.0465
Minimum	2
Maximum	64
Zeros	0
Zeros (%)	0.0%
Memory size	15.8 KiB



Toggle details

Statistics | Histogram | Common values | Extreme values

Quantile statistics

Minimum	2
5-th percentile	5
Q1	16
median	32
Q3	48
95-th percentile	61
Maximum	64
Range	62
Interquartile range (IQR)	32

Descriptive statistics

Standard deviation	18.14571496
Coefficient of variation (CV)	0.5662307882
Kurtosis	-1.21607403
Mean	32.0465
Median Absolute Deviation (MAD)	16
Skewness	0.05788932785
Sum	64093
Variance	329.2669712
Monotocity	Not monotonic

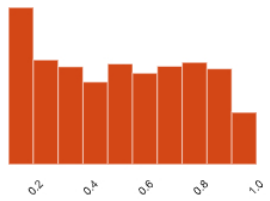
m_dep

Real number ($\mathbb{R}_{\geq 0}$)

Mobile Depth in cm

Distinct	10
Distinct (%)	0.5%
Missing	0
Missing (%)	0.0%
Infinite	0
Infinite (%)	0.0%

Mean	0.50175
Minimum	0.1
Maximum	1
Zeros	0
Zeros (%)	0.0%
Memory size	15.8 KiB



Toggle details

Statistics | Histogram | Common values | Extreme values

Quantile statistics

Minimum	0.1
5-th percentile	0.1
Q1	0.2
median	0.5
Q3	0.8
95-th percentile	1
Maximum	1
Range	0.9
Interquartile range (IQR)	0.6

Descriptive statistics

Standard deviation	0.2884155496
Coefficient of variation (CV)	0.5748192319
Kurtosis	-1.274348884
Mean	0.50175
Median Absolute Deviation (MAD)	0.3
Skewness	0.08908200979
Sum	1003.5
Variance	0.08318352926
Monotocity	Not monotonic

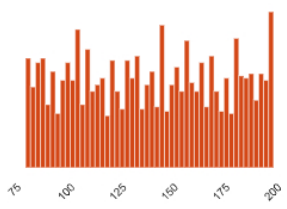
mobile_wt

Real number ($\mathbb{R}_{\geq 0}$)

Weight of mobile phone

Distinct	121
Distinct (%)	6.0%
Missing	0
Missing (%)	0.0%
Infinite	0

Mean	140.249
Minimum	80
Maximum	200
Zeros	0
Zeros (%)	0.0%



Infinite (%)	0.0%	**Memory size**	15.8 KiB

Toggle details

Statistics | Histogram | Common values | Extreme values

Quantile statistics

Minimum	80
5-th percentile	86
Q1	109
median	141
Q3	170
95-th percentile	196
Maximum	200
Range	120
Interquartile range (IQR)	61

Descriptive statistics

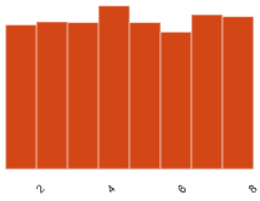
Standard deviation	35.3996549
Coefficient of variation (CV)	0.2524057562
Kurtosis	-1.210376474
Mean	140.249
Median Absolute Deviation (MAD)	31
Skewness	0.006558157429
Sum	280498
Variance	1253.135567
Monotocity	Not monotonic

n_cores

Real number ($R_{\geq 0}$)

Number of cores of processor

Distinct	8	**Mean**	4.5205
Distinct (%)	0.4%	**Minimum**	1
Missing	0	**Maximum**	8
Missing (%)	0.0%	**Zeros**	0
Infinite	0	**Zeros (%)**	0.0%
Infinite (%)	0.0%	**Memory size**	15.8 KiB



Toggle details

Statistics | Histogram | Common values | Extreme values

Quantile statistics

Minimum	1
5-th percentile	1
Q1	3
median	4
Q3	7
95-th percentile	8
Maximum	8
Range	7
Interquartile range (IQR)	4

Descriptive statistics

Standard deviation	2.287836718
Coefficient of variation (CV)	0.5061025811
Kurtosis	-1.229749767
Mean	4.5205
Median Absolute Deviation (MAD)	2
Skewness	0.003627508314
Sum	9041
Variance	5.234196848
Monotocity	Not monotonic

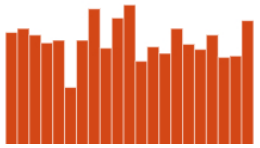
pc

Real number ($R_{\geq 0}$)

ZEROS

Primary Camera mega pixels

Distinct	21	**Mean**	9.9165
Distinct (%)	1.1%	**Minimum**	0
Missing	0	**Maximum**	20
Missing (%)	0.0%	**Zeros**	101



Infinite	0
Infinite (%)	0.0%

Zeros (%)	5.1%
Memory size	15.8 KiB

Toggle details

Statistics | Histogram | Common values | Extreme values

Quantile statistics

Minimum	0
5-th percentile	0
Q1	5
median	10
Q3	15
95-th percentile	20
Maximum	20
Range	20
Interquartile range (IQR)	10

Descriptive statistics

Standard deviation	6.064314941
Coefficient of variation (CV)	0.6115378351
Kurtosis	-1.171498795
Mean	9.9165
Median Absolute Deviation (MAD)	5
Skewness	0.01730615047
Sum	19833
Variance	36.77591571
Monotocity	Not monotonic

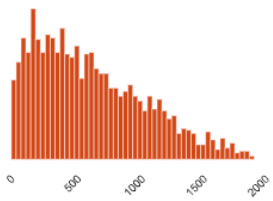
px_height

Real number ($\mathbb{R}_{\geq 0}$)

Pixel Resolution Height

Distinct	1137
Distinct (%)	56.9%
Missing	0
Missing (%)	0.0%
Infinite	0
Infinite (%)	0.0%

Mean	645.108
Minimum	0
Maximum	1960
Zeros	2
Zeros (%)	0.1%
Memory size	15.8 KiB



Toggle details

Statistics | Histogram | Common values | Extreme values

Quantile statistics

Minimum	0
5-th percentile	70.95
Q1	282.75
median	564
Q3	947.25
95-th percentile	1485.05
Maximum	1960
Range	1960
Interquartile range (IQR)	664.5

Descriptive statistics

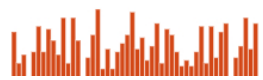
Standard deviation	443.7808108
Coefficient of variation (CV)	0.6879170787
Kurtosis	-0.3158654936
Mean	645.108
Median Absolute Deviation (MAD)	318
Skewness	0.6662712561
Sum	1290216
Variance	196941.408
Monotocity	Not monotonic

px_width

Real number ($\mathbb{R}_{\geq 0}$)

Distinct	1109
Distinct (%)	55.5%

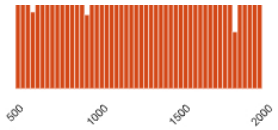
Mean	1251.5155
Minimum	500



Pixel Resolution Width

Missing	0
Missing (%)	0.0%
Infinite	0
Infinite (%)	0.0%

Maximum	1998
Zeros	0
Zeros (%)	0.0%
Memory size	15.8 KiB



Toggle details

Statistics | Histogram | Common values | Extreme values

Quantile statistics

Minimum	500
5-th percentile	579.85
Q1	874.75
median	1247
Q3	1633
95-th percentile	1929.05
Maximum	1998
Range	1498
Interquartile range (IQR)	758.25

Descriptive statistics

Standard deviation	432.1994469
Coefficient of variation (CV)	0.3453408663
Kurtosis	-1.186005229
Mean	1251.5155
Median Absolute Deviation (MAD)	376
Skewness	0.01478747377
Sum	2503031
Variance	186796.3619
Monotocity	Not monotonic

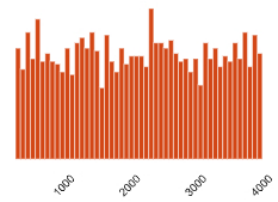
ram

Real number ($\mathbb{R}_{\geq 0}$)

Random Access Memory in Mega Bytes

Distinct	1562
Distinct (%)	78.1%
Missing	0
Missing (%)	0.0%
Infinite	0
Infinite (%)	0.0%

Mean	2124.213
Minimum	256
Maximum	3998
Zeros	0
Zeros (%)	0.0%
Memory size	15.8 KiB



Toggle details

Statistics | Histogram | Common values | Extreme values

Quantile statistics

Minimum	256
5-th percentile	445
Q1	1207.5
median	2146.5
Q3	3064.5
95-th percentile	3826.35
Maximum	3998
Range	3742
Interquartile range (IQR)	1857

Descriptive statistics

Standard deviation	1084.732044
Coefficient of variation (CV)	0.5106512594
Kurtosis	-1.19191307
Mean	2124.213
Median Absolute Deviation (MAD)	932.5
Skewness	0.006628035399
Sum	4248426
Variance	1176643.606
Monotocity	Not monotonic

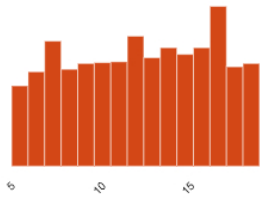
sc_h

Real number ($\mathbb{R}_{\geq 0}$)

Screen Height of mobile in cm

Distinct	15
Distinct (%)	0.8%
Missing	0
Missing (%)	0.0%
Infinite	0
Infinite (%)	0.0%

Mean	12.3065
Minimum	5
Maximum	19
Zeros	0
Zeros (%)	0.0%
Memory size	15.8 KiB



Toggle details

Statistics | Histogram | Common values | Extreme values

Quantile statistics

Minimum	5
5-th percentile	6
Q1	9
median	12
Q3	16
95-th percentile	19
Maximum	19
Range	14
Interquartile range (IQR)	7

Descriptive statistics

Standard deviation	4.213245004
Coefficient of variation (CV)	0.3423593227
Kurtosis	-1.190791247
Mean	12.3065
Median Absolute Deviation (MAD)	4
Skewness	-0.09888424098
Sum	24613
Variance	17.75143347
Monotocity	Not monotonic

SC_W

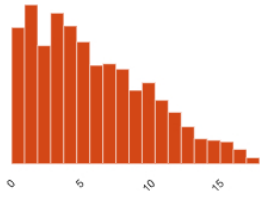
Real number ($\mathbb{R}_{\geq 0}$)

ZEROS

Screen Width of mobile in cm

Distinct	19
Distinct (%)	0.9%
Missing	0
Missing (%)	0.0%
Infinite	0
Infinite (%)	0.0%

Mean	5.767
Minimum	0
Maximum	18
Zeros	180
Zeros (%)	9.0%
Memory size	15.8 KiB



Toggle details

Statistics | Histogram | Common values | Extreme values

Quantile statistics

Minimum	0
5-th percentile	0
Q1	2
median	5
Q3	9
95-th percentile	14
Maximum	18
Range	18
Interquartile range (IQR)	7

Descriptive statistics

Standard deviation	4.356397606
Coefficient of variation (CV)	0.7554010067
Kurtosis	-0.3895227894
Mean	5.767
Median Absolute Deviation (MAD)	3
Skewness	0.6337870734
Sum	11534
Variance	18.9782001
Monotocity	Not monotonic

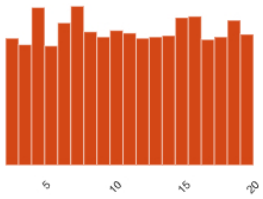
talk_time

Real number ($R_{\geq 0}$)

longest time that a single battery charge will last when you are

Distinct	19
Distinct (%)	0.9%
Missing	0
Missing (%)	0.0%
Infinite	0
Infinite (%)	0.0%

Mean	11.011
Minimum	2
Maximum	20
Zeros	0
Zeros (%)	0.0%
Memory size	15.8 KiB



Toggle details

Statistics | Histogram | Common values | Extreme values

Quantile statistics

Minimum	2
5-th percentile	3
Q1	6
median	11
Q3	16
95-th percentile	20
Maximum	20
Range	18
Interquartile range (IQR)	10

Descriptive statistics

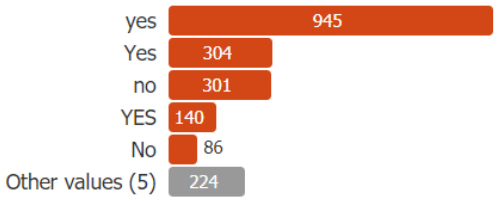
Standard deviation	5.463955198
Coefficient of variation (CV)	0.4962269728
Kurtosis	-1.218590963
Mean	11.011
Median Absolute Deviation (MAD)	5
Skewness	0.009511762222
Sum	22022
Variance	29.8548064
Monotocity	Not monotonic

three_g

Categorical

Has 3G or not

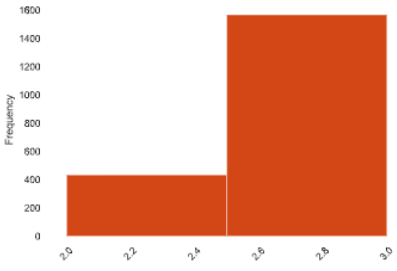
Distinct	10
Distinct (%)	0.5%
Missing	0
Missing (%)	0.0%
Memory size	116.9 KiB



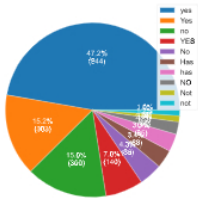
Toggle details

Overview | Categories | Words | Characters

Value	Count	Frequency (%)
yes	945	47.2%
Yes	304	15.2%
no	301	15.0%
YES	140	7.0%
No	86	4.3%
Has	68	3.4%
has	66	3.3%
NO	46	2.3%
Not	23	1.1%



Histogram of lengths of the category



not	21		1.1%

touch_screen

Categorical

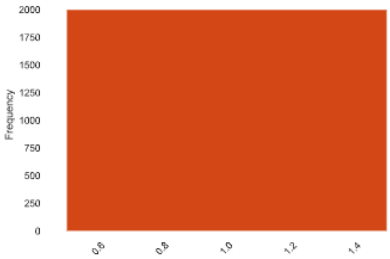
Has touch screen or not, 1 = yes, 0 = no

Distinct	2
Distinct (%)	0.1%
Missing	0
Missing (%)	0.0%
Memory size	113.4 KiB

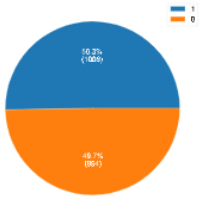
Toggle details

Overview | Categories | Words | Characters

Value	Count	Frequency (%)
1	1006	50.3%
0	994	49.7%



Histogram of lengths of the category

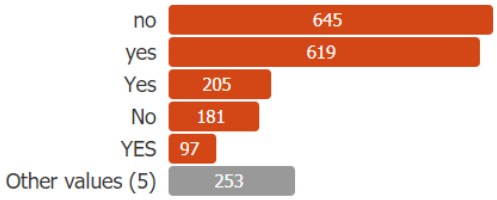


wifi

Categorical

Has wifi or not

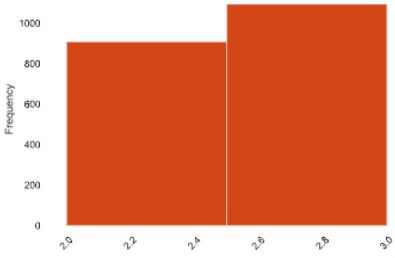
Distinct	10
Distinct (%)	0.5%
Missing	0
Missing (%)	0.0%
Memory size	116.4 KiB



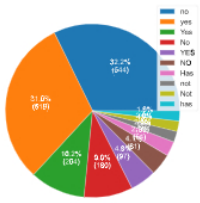
Toggle details

Overview | Categories | Words | Characters

Value	Count	Frequency (%)
no	645	32.2%
yes	619	30.9%
Yes	205	10.2%
No	181	9.0%
YES	97	4.9%
NO	81	4.0%
Has	55	2.8%
not	40	2.0%
Not	39	1.9%
has	38	1.9%



Histogram of lengths of the category



price_category
Categorical

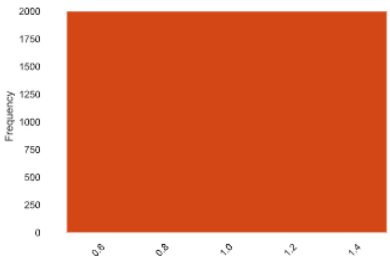
This is the target variable with indicating if the mobile phone got a high price. 1 = yes, 0 = no

Distinct	2
Distinct (%)	0.1%
Missing	0
Missing (%)	0.0%
Memory size	113.4 KiB

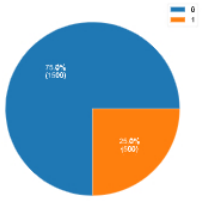
Toggle details

Overview | Categories | Words | Characters

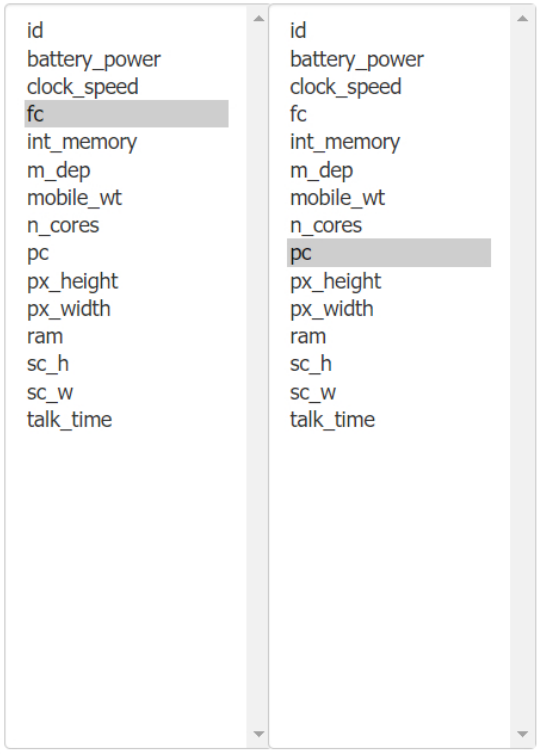
Value	Count	Frequency (%)
0	1500	75.0%
1	500	25.0%

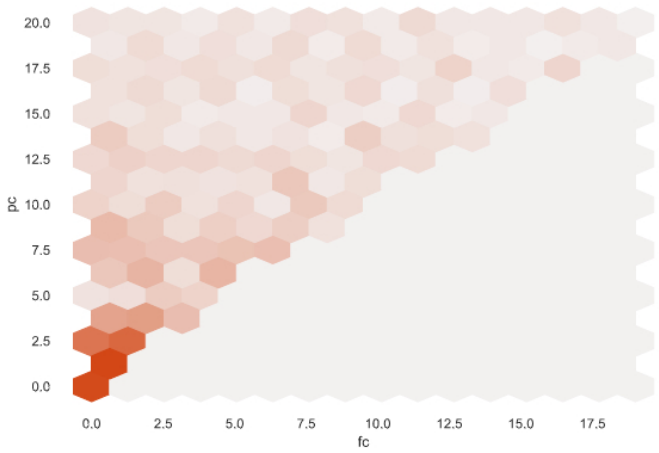


Histogram of lengths of the category



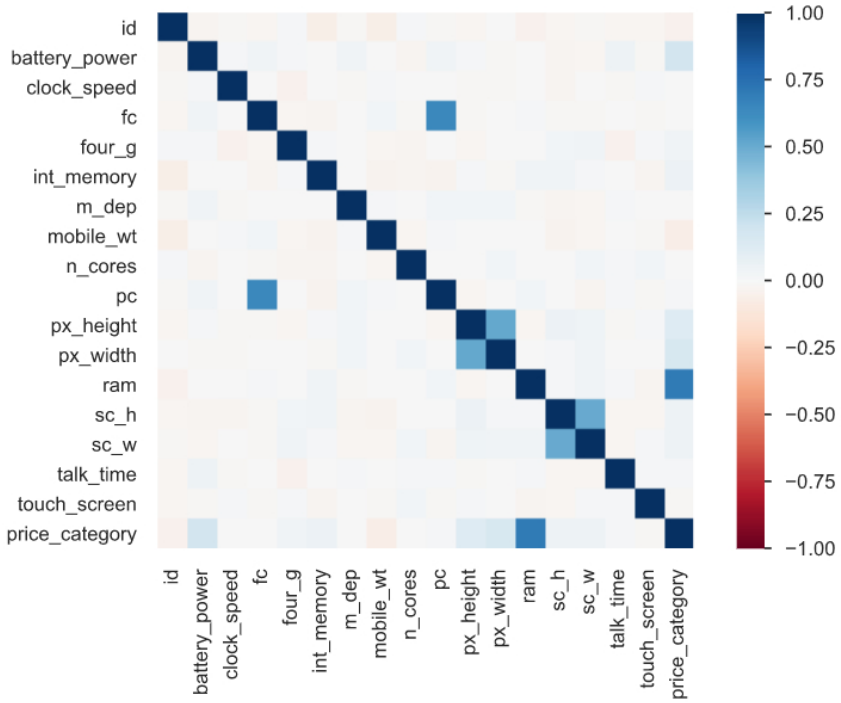
Interactions





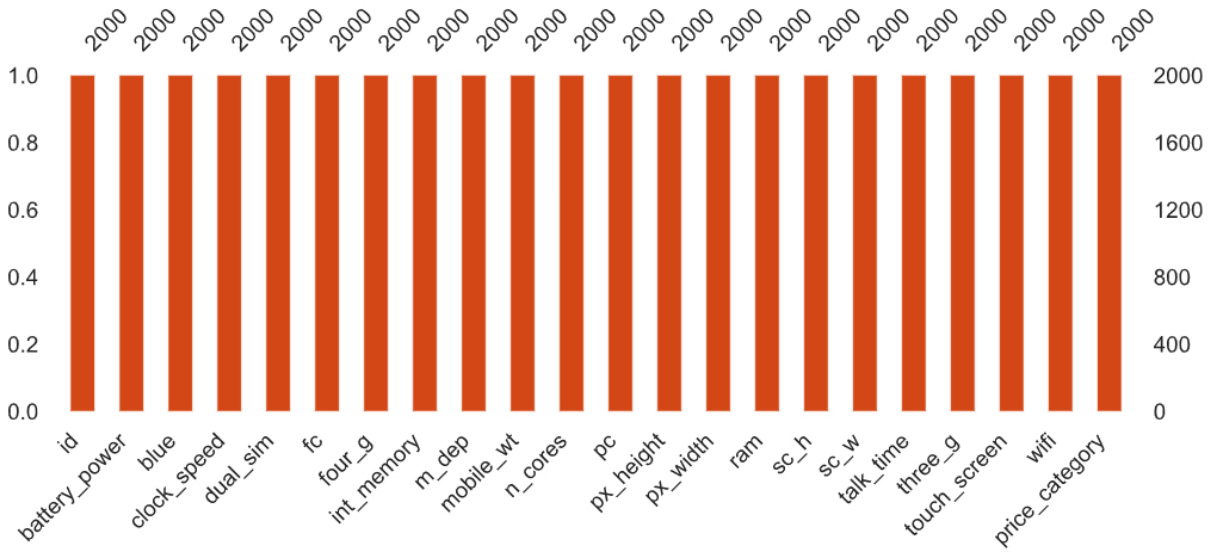
Correlations

Pearson's r | Spearman's ρ | Kendall's τ | Phik (φk) | Cramér's V (φc) | Toggle correlation descriptions



Missing values

Count | Matrix



A simple visualization of nullity by column.

Sample

First rows

	id	battery_power	blue	clock_speed	dual_sim	fc	four_g	int_memory	m_dep	mobile_wt	n_cores	pc	px_h
0	0	842	no	2.2	no	1	0	7	0.6	188	2	2	20
1	1	1021	yes	0.5	YES	0	1	53	0.7	136	3	6	905
2	2	563	yes	0.5	Yes	2	1	41	0.9	145	5	6	1263
3	3	615	has	2.5	no	0	0	10	0.8	131	6	9	1216
4	4	1821	yes	1.2	NO	13	1	44	0.6	141	2	14	1208
5	5	1859	No	0.5	yes	3	0	22	0.7	164	1	7	1004
6	6	1821	no	1.7	NO	4	1	10	0.8	139	8	10	381
7	7	1954	no	0.5	Yes	0	0	24	0.8	187	4	0	512
8	8	1445	yes	0.5	NO	0	0	53	0.7	174	7	14	386
9	9	509	yes	0.6	has	2	1	9	0.1	93	5	15	1137

Last rows

	id	battery_power	blue	clock_speed	dual_sim	fc	four_g	int_memory	m_dep	mobile_wt	n_cores	pc
1990	1990	1617	yes	2.4	no	8	1	36	0.8	85	1	9
1991	1991	1882	no	2.0	no	11	1	44	0.8	113	8	19
1992	1992	674	has	2.9	yes	1	0	21	0.2	198	3	4
1993	1993	1467	yes	0.5	no	0	0	18	0.6	122	5	0
1994	1994	858	no	2.2	no	1	0	50	0.1	84	1	2
1995	1995	794	yes	0.5	yes	0	1	2	0.8	106	6	14
1996	1996	1965	yes	2.6	yes	0	0	39	0.2	187	4	3
1997	1997	1911	no	0.9	yes	1	1	36	0.7	108	8	3
1998	1998	1512	no	0.9	not	4	1	46	0.1	145	5	5
1999	1999	510	Yes	2.0	yes	5	1	45	0.9	168	6	16

Overview

Overview | Variables | Warnings 5 | Reproduction

Dataset statistics

Number of variables	22
Number of observations	2000
Missing cells	0
Missing cells (%)	0.0%
Duplicate rows	0
Duplicate rows (%)	0.0%
Total size in memory	248.2 KiB
Average record size in memory	127.1 B

Variable types

Numeric	15
Boolean	7

Variables

id

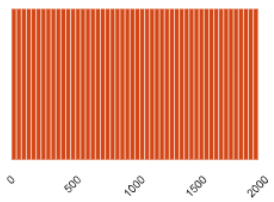
Real number ($\mathbb{R}_{\geq 0}$)

`UNIFORM`
`UNIQUE`

ID

Distinct	2000
Distinct (%)	100.0%
Missing	0
Missing (%)	0.0%
Infinite	0
Infinite (%)	0.0%

Mean	999.5
Minimum	0
Maximum	1999
Zeros	1
Zeros (%)	< 0.1%
Memory size	15.8 KiB



Toggle details

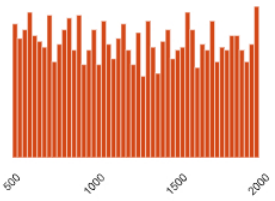
battery_power

Real number ($\mathbb{R}_{\geq 0}$)

Total energy a battery can store in one time measured in mAh

Distinct	1094
Distinct (%)	54.7%
Missing	0
Missing (%)	0.0%
Infinite	0
Infinite (%)	0.0%

Mean	1238.5185
Minimum	501
Maximum	1998
Zeros	0
Zeros (%)	0.0%
Memory size	15.8 KiB



Toggle details

Statistics | Histogram | Common values | Extreme values

Quantile statistics

Descriptive statistics

Minimum	501	**Standard deviation**	439.4182061
5-th percentile	570.95	**Coefficient of variation (CV)**	0.3547934133
Q1	851.75	**Kurtosis**	-1.224143883
median	1226	**Mean**	1238.5185
Q3	1615.25	**Median Absolute Deviation (MAD)**	382
95-th percentile	1930.15	**Skewness**	0.03189847179
Maximum	1998	**Sum**	2477037
Range	1497	**Variance**	193088.3598
Interquartile range (IQR)	763.5	**Monotocity**	Not monotonic

has_bluetooth

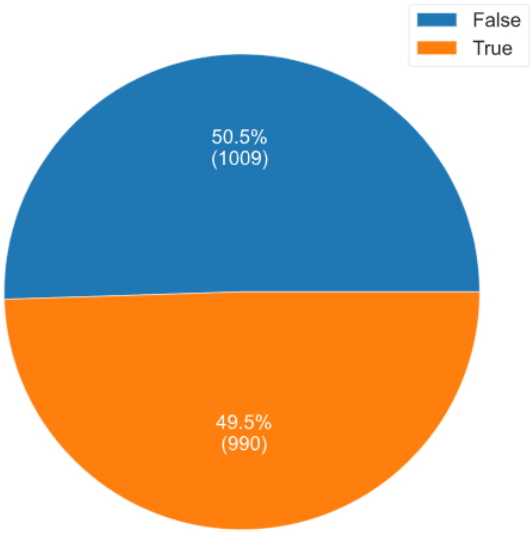
Boolean

Has bluetooth or not

Distinct	2
Distinct (%)	0.1%
Missing	0
Missing (%)	0.0%
Memory size	2.1 KiB

Toggle details

Common Values | Chart



clock_speed

Real number ($R_{>0}$)

speed at which microprocessor executes instructions

Distinct	26
Distinct (%)	1.3%
Missing	0
Missing (%)	0.0%

Mean	1.52225
Minimum	0.5
Maximum	3
Zeros	0

Infinite	0	**Zeros (%)**	0.0%
Infinite (%)	0.0%	**Memory size**	15.8 KiB

Toggle details

Statistics | Histogram | Common values | Extreme values

Quantile statistics

Minimum	0.5
5-th percentile	0.5
Q1	0.7
median	1.5
Q3	2.2
95-th percentile	2.8
Maximum	3
Range	2.5
Interquartile range (IQR)	1.5

Descriptive statistics

Standard deviation	0.8160042089
Coefficient of variation (CV)	0.5360513772
Kurtosis	-1.323417222
Mean	1.52225
Median Absolute Deviation (MAD)	0.8
Skewness	0.1780841203
Sum	3044.5
Variance	0.6658628689
Monotocity	Not monotonic

has_dual_sim

Boolean

Has dual sim support or not

Distinct	2
Distinct (%)	0.1%
Missing	0
Missing (%)	0.0%
Memory size	2.1 KiB

Toggle details

Common Values | Chart

Value	Count	Frequency (%)
True	1019	50.9%
False	981	49.0%

front_cam_resolution

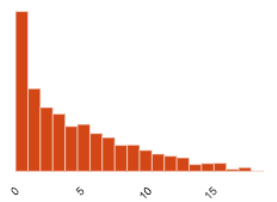
Real number ($R_{\geq 0}$)

ZEROS

Front Camera mega pixels

Distinct	20
Distinct (%)	1.0%
Missing	0
Missing (%)	0.0%
Infinite	0
Infinite (%)	0.0%

Mean	4.3095
Minimum	0
Maximum	19
Zeros	474
Zeros (%)	23.7%
Memory size	15.8 KiB



Toggle details

Statistics | Histogram | Common values | Extreme values

Quantile statistics	
Minimum	0
5-th percentile	0
Q1	1
median	3
Q3	7
95-th percentile	13
Maximum	19
Range	19
Interquartile range (IQR)	6

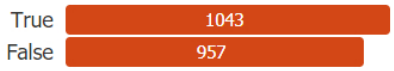
Descriptive statistics	
Standard deviation	4.341443748
Coefficient of variation (CV)	1.007412402
Kurtosis	0.2770763246
Mean	4.3095
Median Absolute Deviation (MAD)	3
Skewness	1.019811411
Sum	8619
Variance	18.84813382
Monotocity	Not monotonic

has_four_g

Boolean

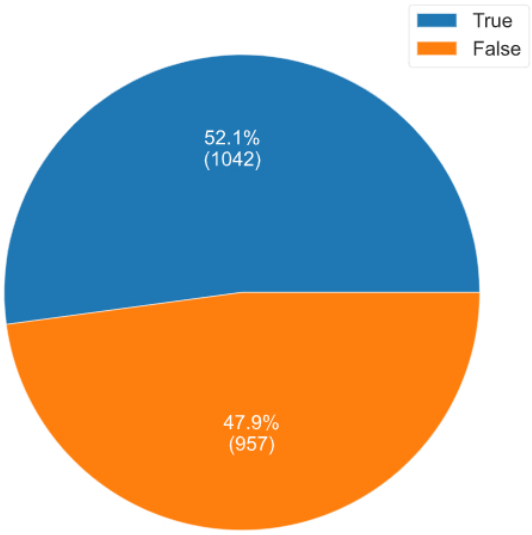
Has 4G or not. 1 = yes , 0 = no

Distinct	2
Distinct (%)	0.1%
Missing	0
Missing (%)	0.0%
Memory size	2.1 KiB



Toggle details

Common Values | Chart



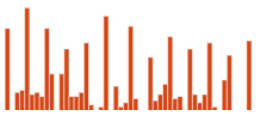
int_memory

Real number ($\mathbb{R}_{>0}$)

internal Memory in Gigabytes

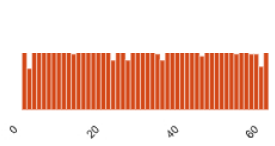
Distinct	63
Distinct (%)	3.1%
Missing	0

Mean	32.0465
Minimum	2
Maximum	64



Internal Memory in Gigabytes

Missing (%)	0.0%	**Zeros**	0
Infinite	0	**Zeros (%)**	0.0%
Infinite (%)	0.0%	**Memory size**	15.8 KiB



Toggle details

Statistics | Histogram | Common values | Extreme values

Quantile statistics

Minimum	2
5-th percentile	5
Q1	16
median	32
Q3	48
95-th percentile	61
Maximum	64
Range	62
Interquartile range (IQR)	32

Descriptive statistics

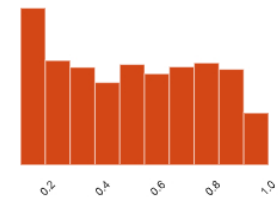
Standard deviation	18.14571496
Coefficient of variation (CV)	0.5662307882
Kurtosis	-1.21607403
Mean	32.0465
Median Absolute Deviation (MAD)	16
Skewness	0.05788932785
Sum	64093
Variance	329.2669712
Monotocity	Not monotonic

mobile_depth

Real number ($R_{\geq 0}$)

Mobile Depth in cm

Distinct	10	**Mean**	0.50175
Distinct (%)	0.5%	**Minimum**	0.1
Missing	0	**Maximum**	1
Missing (%)	0.0%	**Zeros**	0
Infinite	0	**Zeros (%)**	0.0%
Infinite (%)	0.0%	**Memory size**	15.8 KiB



Toggle details

Statistics | Histogram | Common values | Extreme values

Quantile statistics

Minimum	0.1
5-th percentile	0.1
Q1	0.2
median	0.5
Q3	0.8
95-th percentile	1
Maximum	1
Range	0.9
Interquartile range (IQR)	0.6

Descriptive statistics

Standard deviation	0.2884155496
Coefficient of variation (CV)	0.5748192319
Kurtosis	-1.274348884
Mean	0.50175
Median Absolute Deviation (MAD)	0.3
Skewness	0.08908200979
Sum	1003.5
Variance	0.08318352926
Monotocity	Not monotonic

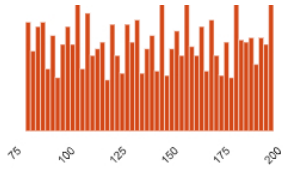
mobile_weight

Distinct	121	**Mean**	140.249

Real number ($R_{\geq 0}$)

Weight of mobile phone

Distinct (%)	6.0%	**Minimum**	80
Missing	0	**Maximum**	200
Missing (%)	0.0%	**Zeros**	0
Infinite	0	**Zeros (%)**	0.0%
Infinite (%)	0.0%	**Memory size**	15.8 KiB



Toggle details

Statistics | Histogram | Common values | Extreme values

Quantile statistics

Minimum	80
5-th percentile	86
Q1	109
median	141
Q3	170
95-th percentile	196
Maximum	200
Range	120
Interquartile range (IQR)	61

Descriptive statistics

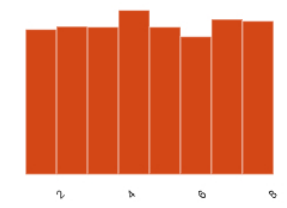
Standard deviation	35.3996549
Coefficient of variation (CV)	0.2524057562
Kurtosis	-1.210376474
Mean	140.249
Median Absolute Deviation (MAD)	31
Skewness	0.006558157429
Sum	280498
Variance	1253.135567
Monotocity	Not monotonic

number_of_cores

Real number ($R_{\geq 0}$)

Number of cores of processor

Distinct	8	**Mean**	4.5205
Distinct (%)	0.4%	**Minimum**	1
Missing	0	**Maximum**	8
Missing (%)	0.0%	**Zeros**	0
Infinite	0	**Zeros (%)**	0.0%
Infinite (%)	0.0%	**Memory size**	15.8 KiB



Toggle details

Statistics | Histogram | Common values | Extreme values

Quantile statistics

Minimum	1
5-th percentile	1
Q1	3
median	4
Q3	7
95-th percentile	8
Maximum	8
Range	7
Interquartile range (IQR)	4

Descriptive statistics

Standard deviation	2.287836718
Coefficient of variation (CV)	0.5061025811
Kurtosis	-1.229749767
Mean	4.5205
Median Absolute Deviation (MAD)	2
Skewness	0.003627508314
Sum	9041
Variance	5.234196848
Monotocity	Not monotonic

primary_cam_resolut...

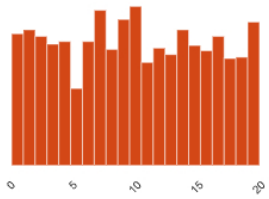
Real number ($R_{>0}$)

ZEROS

Primary Camera mega pixels

Distinct	21
Distinct (%)	1.1%
Missing	0
Missing (%)	0.0%
Infinite	0
Infinite (%)	0.0%

Mean	9.9165
Minimum	0
Maximum	20
Zeros	101
Zeros (%)	5.1%
Memory size	15.8 KiB



Toggle details

Statistics | Histogram | Common values | Extreme values

Quantile statistics

Minimum	0
5-th percentile	0
Q1	5
median	10
Q3	15
95-th percentile	20
Maximum	20
Range	20
Interquartile range (IQR)	10

Descriptive statistics

Standard deviation	6.064314941
Coefficient of variation (CV)	0.6115378351
Kurtosis	-1.171498795
Mean	9.9165
Median Absolute Deviation (MAD)	5
Skewness	0.01730615047
Sum	19833
Variance	36.77591571
Monotocity	Not monotonic

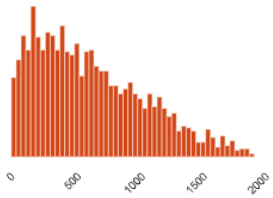
px_height

Real number ($R_{\geq 0}$)

Pixel Resolution Height

Distinct	1137
Distinct (%)	56.9%
Missing	0
Missing (%)	0.0%
Infinite	0
Infinite (%)	0.0%

Mean	645.108
Minimum	0
Maximum	1960
Zeros	2
Zeros (%)	0.1%
Memory size	15.8 KiB



Toggle details

Statistics | Histogram | Common values | Extreme values

Quantile statistics

Minimum	0
5-th percentile	70.95
Q1	282.75
median	564
Q3	947.25
95-th percentile	1485.05
Maximum	1960
Range	1960
Interquartile range (IQR)	664.5

Descriptive statistics

Standard deviation	443.7808108
Coefficient of variation (CV)	0.6879170787
Kurtosis	-0.3158654936
Mean	645.108
Median Absolute Deviation (MAD)	318
Skewness	0.6662712561
Sum	1290216
Variance	196941.408
Monotocity	Not monotonic

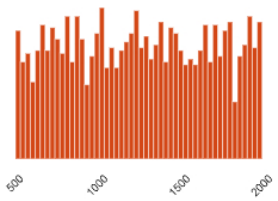
px_width

Real number ($\mathbb{R}_{\geq 0}$)

Pixel Resolution Width

Distinct	1109
Distinct (%)	55.5%
Missing	0
Missing (%)	0.0%
Infinite	0
Infinite (%)	0.0%

Mean	1251.5155
Minimum	500
Maximum	1998
Zeros	0
Zeros (%)	0.0%
Memory size	15.8 KiB



Toggle details

Statistics | Histogram | Common values | Extreme values

Quantile statistics

Minimum	500
5-th percentile	579.85
Q1	874.75
median	1247
Q3	1633
95-th percentile	1929.05
Maximum	1998
Range	1498
Interquartile range (IQR)	758.25

Descriptive statistics

Standard deviation	432.1994469
Coefficient of variation (CV)	0.3453408663
Kurtosis	-1.186005229
Mean	1251.5155
Median Absolute Deviation (MAD)	376
Skewness	0.01478747377
Sum	2503031
Variance	186796.3619
Monotocity	Not monotonic

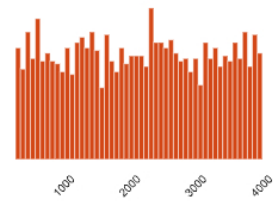
ram

Real number ($\mathbb{R}_{\geq 0}$)

Random Access Memory in Mega Bytes

Distinct	1562
Distinct (%)	78.1%
Missing	0
Missing (%)	0.0%
Infinite	0
Infinite (%)	0.0%

Mean	2124.213
Minimum	256
Maximum	3998
Zeros	0
Zeros (%)	0.0%
Memory size	15.8 KiB



Toggle details

Statistics | Histogram | Common values | Extreme values

Quantile statistics

Minimum	256
5-th percentile	445
Q1	1207.5
median	2146.5
Q3	3064.5
95-th percentile	3826.35
Maximum	3998
Range	3742
Interquartile range (IQR)	1857

Descriptive statistics

Standard deviation	1084.732044
Coefficient of variation (CV)	0.5106512594
Kurtosis	-1.19191307
Mean	2124.213
Median Absolute Deviation (MAD)	932.5
Skewness	0.006628035399
Sum	4248426
Variance	1176643.606
Monotocity	Not monotonic

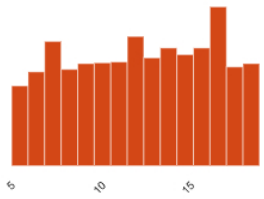
screen_height

Real number ($\mathbb{R}_{>0}$)

Screen Height of mobile in cm

Distinct	15
Distinct (%)	0.8%
Missing	0
Missing (%)	0.0%
Infinite	0
Infinite (%)	0.0%

Mean	12.3065
Minimum	5
Maximum	19
Zeros	0
Zeros (%)	0.0%
Memory size	15.8 KiB



Toggle details

Statistics | Histogram | Common values | Extreme values

Quantile statistics

Minimum	5
5-th percentile	6
Q1	9
median	12
Q3	16
95-th percentile	19
Maximum	19
Range	14
Interquartile range (IQR)	7

Descriptive statistics

Standard deviation	4.213245004
Coefficient of variation (CV)	0.3423593227
Kurtosis	-1.190791247
Mean	12.3065
Median Absolute Deviation (MAD)	4
Skewness	-0.09888424098
Sum	24613
Variance	17.75143347
Monotocity	Not monotonic

screen_width

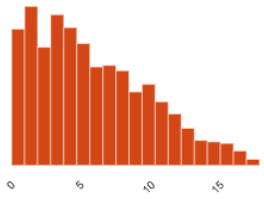
Real number ($\mathbb{R}_{\geq 0}$)

ZEROS

Screen Width of mobile in cm

Distinct	19
Distinct (%)	0.9%
Missing	0
Missing (%)	0.0%
Infinite	0
Infinite (%)	0.0%

Mean	5.767
Minimum	0
Maximum	18
Zeros	180
Zeros (%)	9.0%
Memory size	15.8 KiB



Toggle details

Statistics | Histogram | Common values | Extreme values

Quantile statistics

Minimum	0
5-th percentile	0
Q1	2
median	5
Q3	9
95-th percentile	14
Maximum	18

Descriptive statistics

Standard deviation	4.356397606
Coefficient of variation (CV)	0.7554010067
Kurtosis	-0.3895227894
Mean	5.767
Median Absolute Deviation (MAD)	3
Skewness	0.6337870734
Sum	11534

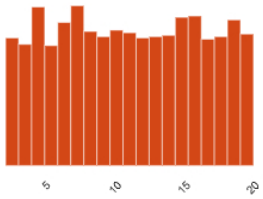
Range	18	**Variance**	18.9782001
Interquartile range (IQR)	7	**Monotocity**	Not monotonic

talk_time

Real number ($R_{\geq 0}$)

longest time that a single battery charge will last when you are

Distinct	19	**Mean**	11.011
Distinct (%)	0.9%	**Minimum**	2
Missing	0	**Maximum**	20
Missing (%)	0.0%	**Zeros**	0
Infinite	0	**Zeros (%)**	0.0%
Infinite (%)	0.0%	**Memory size**	15.8 KiB



Toggle details

Statistics | Histogram | Common values | Extreme values

Quantile statistics

Minimum	2
5-th percentile	3
Q1	6
median	11
Q3	16
95-th percentile	20
Maximum	20
Range	18
Interquartile range (IQR)	10

Descriptive statistics

Standard deviation	5.463955198
Coefficient of variation (CV)	0.4962269728
Kurtosis	-1.218590963
Mean	11.011
Median Absolute Deviation (MAD)	5
Skewness	0.009511762222
Sum	22022
Variance	29.8548064
Monotocity	Not monotonic

has_three_g

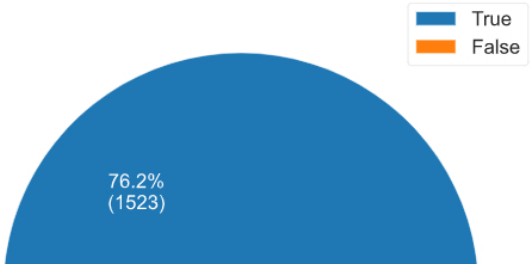
Boolean

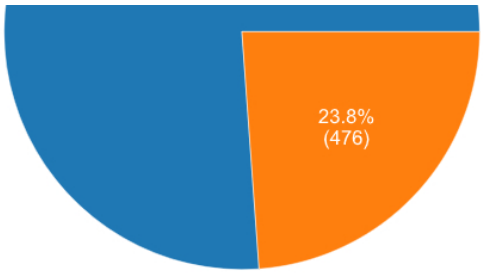
Has 3G or not

Distinct	2		
Distinct (%)	0.1%		
Missing	0		
Missing (%)	0.0%		
Memory size	2.1 KiB		

Toggle details

Common Values | Chart





has_touch_screen

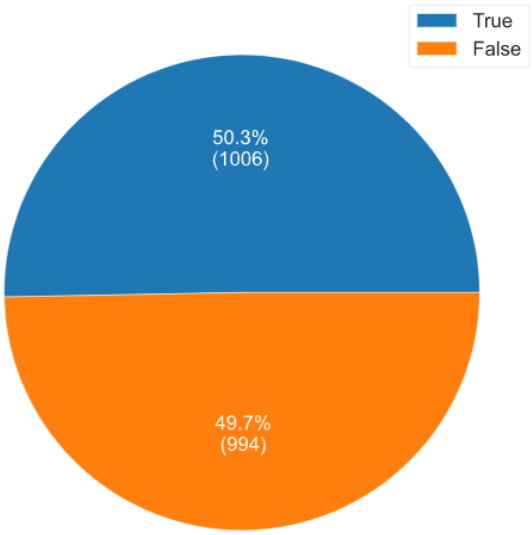
Boolean

Has touch screen or not, 1 = yes, 0 = no

Distinct	2
Distinct (%)	0.1%
Missing	0
Missing (%)	0.0%
Memory size	2.1 KiB

Toggle details

Common Values | Chart



has_wifi

Boolean

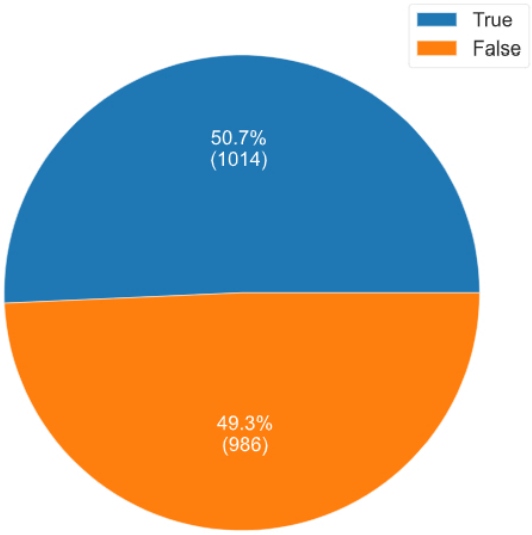
Has wifi or not

Distinct	2
Distinct (%)	0.1%
Missing	0
Missing (%)	0.0%

Memory size	2.1 KiB

Toggle details

Common Values | Chart



is_expensive

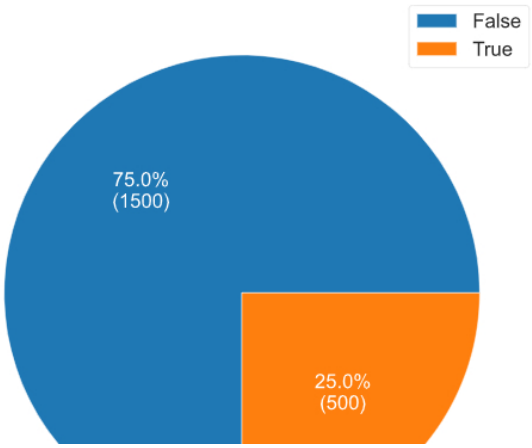
Boolean

This is the target variable with indicating if the mobile phone got a high price. 1 = yes, 0 = no

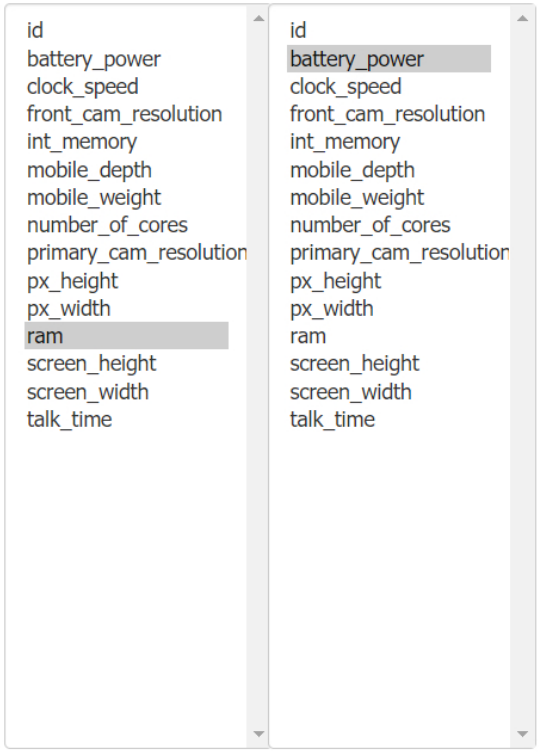
Distinct	2
Distinct (%)	0.1%
Missing	0
Missing (%)	0.0%
Memory size	2.1 KiB

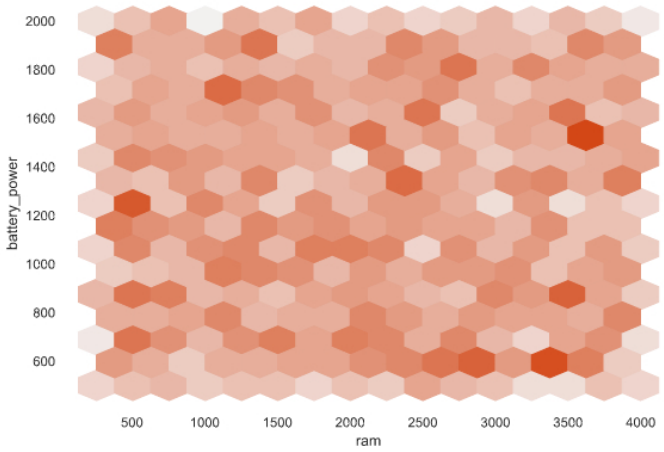
Toggle details

Common Values | Chart

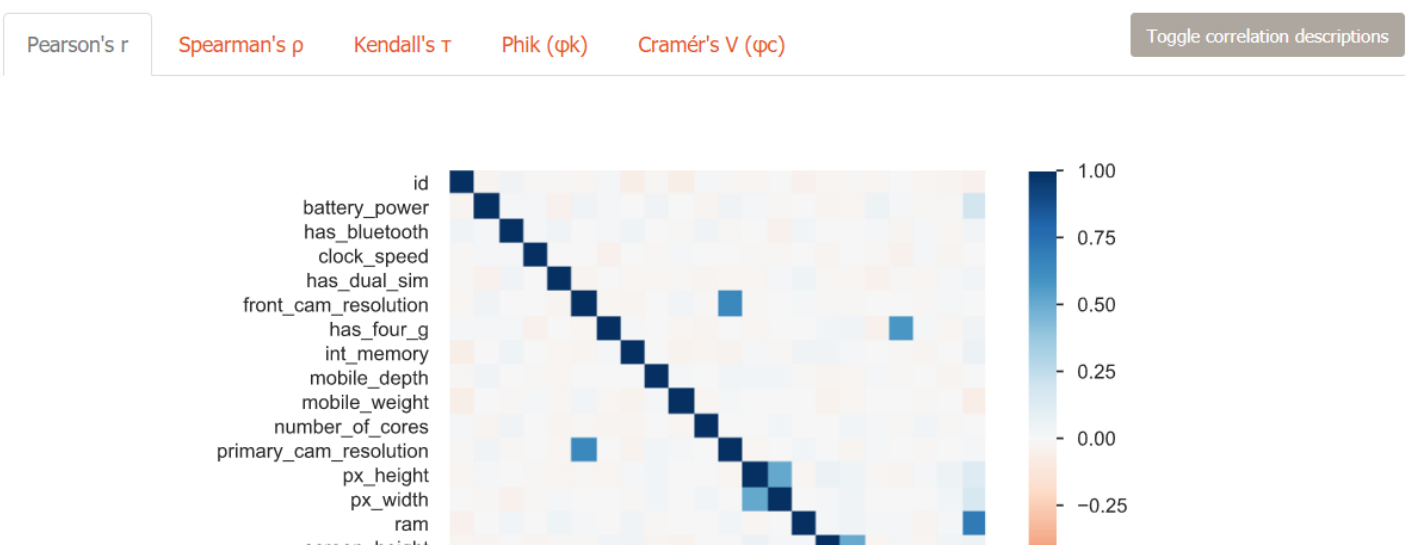


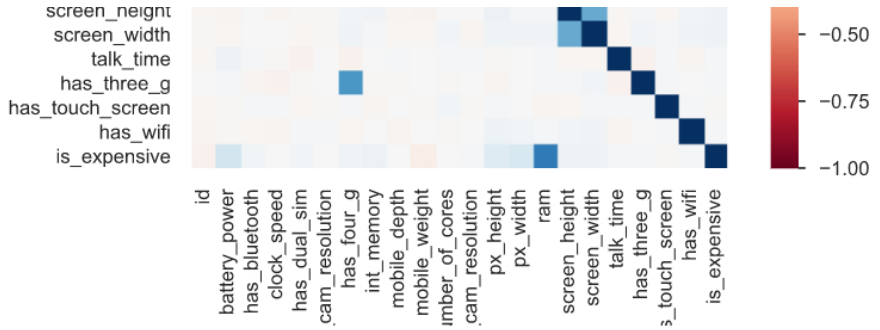
Interactions





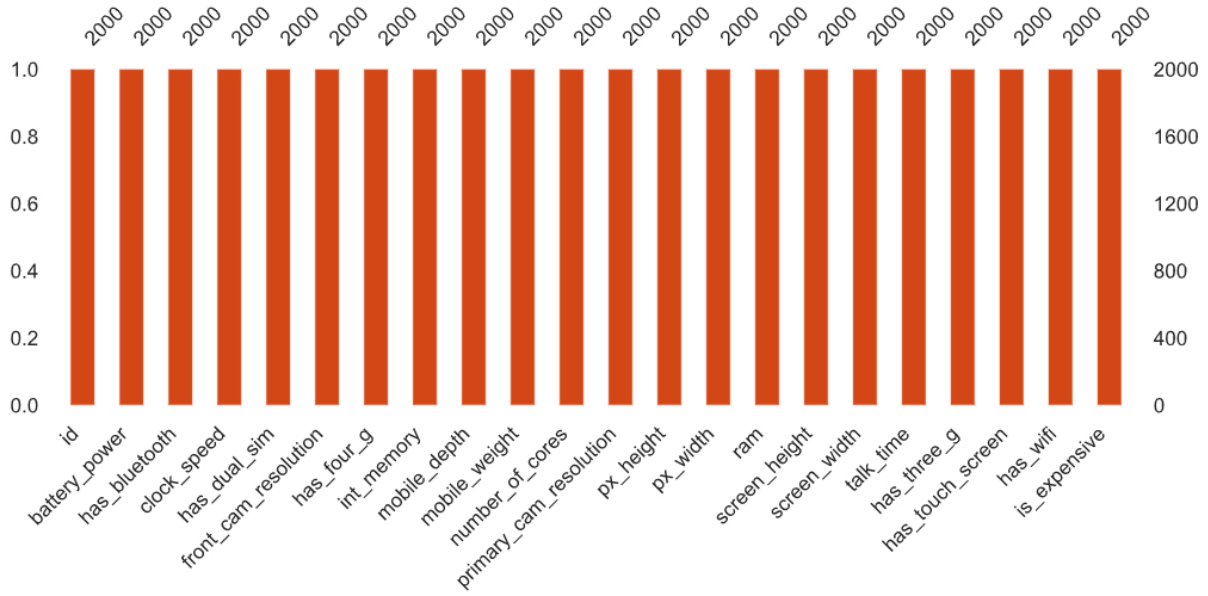
Correlations





Missing values

Count | Matrix



A simple visualization of nullity by column.

Sample

First rows

	id	battery_power	has_bluetooth	clock_speed	has_dual_sim	front_cam_resolution	has_four_g	int_memory	mo
0	0	842	False	2.2	False	1	False	7	0.6
1	1	1021	True	0.5	True	0	True	53	0.7
2	2	563	True	0.5	True	2	True	41	0.9
3	3	615	True	2.5	False	0	False	10	0.8
4	4	1821	True	1.2	False	13	True	44	0.6

5	5	1859	False	0.5	True	3	False	22	0.7
6	6	1821	False	1.7	False	4	True	10	0.8
7	7	1954	False	0.5	True	0	False	24	0.8
8	8	1445	True	0.5	False	0	False	53	0.7
9	9	509	True	0.6	True	2	True	9	0.1

Last rows

	id	battery_power	has_bluetooth	clock_speed	has_dual_sim	front_cam_resolution	has_four_g	int_memory
1990	1990	1617	True	2.4	False	8	True	36
1991	1991	1882	False	2.0	False	11	True	44
1992	1992	674	True	2.9	True	1	False	21
1993	1993	1467	True	0.5	False	0	False	18
1994	1994	858	False	2.2	False	1	False	50
1995	1995	794	True	0.5	True	0	True	2
1996	1996	1965	True	2.6	True	0	False	39
1997	1997	1911	False	0.9	True	1	True	36
1998	1998	1512	False	0.9	False	4	True	46
1999	1999	510	True	2.0	True	5	True	45

Overview

Overview | Variables | Warnings 2 | Reproduction

Dataset statistics

Number of variables	21
Number of observations	2000
Missing cells	0
Missing cells (%)	0.0%
Duplicate rows	0
Duplicate rows (%)	0.0%
Total size in memory	232.6 KiB
Average record size in memory	119.1 B

Variable types

Categorical	11
Boolean	7
Unsupported	2
Numeric	1

Variables

battery_power_categ...

Categorical

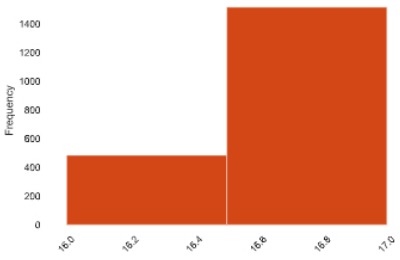
Distinct	4
Distinct (%)	0.2%
Missing	0
Missing (%)	0.0%
Memory size	2.3 KiB

(499.502, 875.25] 537
(1623.75, 1998.0] 490
(1249.5, 1623.75] 489
(875.25, 1249.5] 484

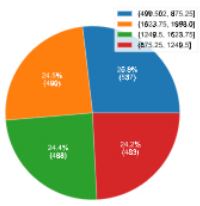
Toggle details

Overview | Categories | Words | Characters

Value	Count	Frequency (%)
(499.502...	537	26.9%
(1623.75...	490	24.5%
(1249.5, ...	489	24.4%
(875.25, ...	484	24.2%



Histogram of lengths of the category



has_bluetooth

Boolean

Distinct	2
Distinct (%)	0.1%

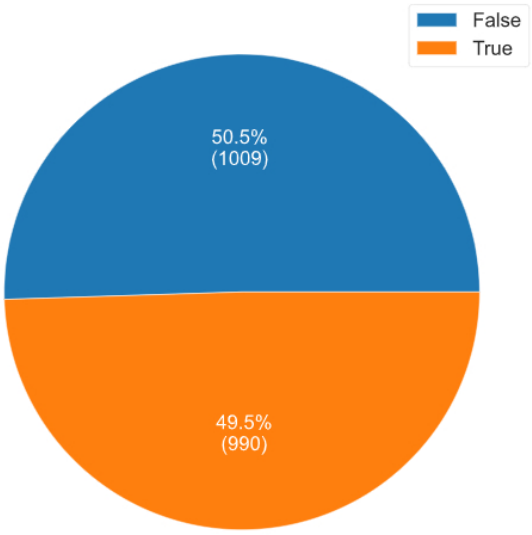
False 1010
True 990

Has bluetooth or not

Missing	0
Missing (%)	0.0%
Memory size	2.1 KiB

Toggle details

Common Values | Chart



clock_speed_category

Categorical

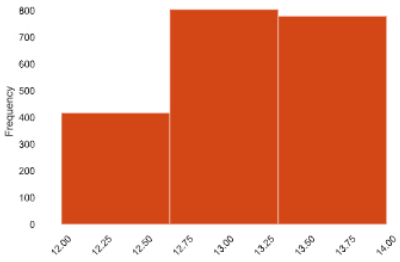
Distinct	4
Distinct (%)	0.2%
Missing	0
Missing (%)	0.0%
Memory size	2.3 KiB

(0.497, 1.125]	779
(2.375, 3.0]	417
(1.75, 2.375]	407
(1.125, 1.75]	397

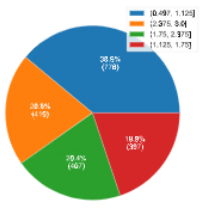
Toggle details

Overview | Categories | Words | Characters

Value	Count	Frequency (%)
(0.497, 1...	779	39.0%
(2.375, 3...	417	20.8%
(1.75, 2....	407	20.3%
(1.125, 1...	397	19.9%



Histogram of lengths of the category



Distinct	2

True	1019

has_dual_sim

Boolean

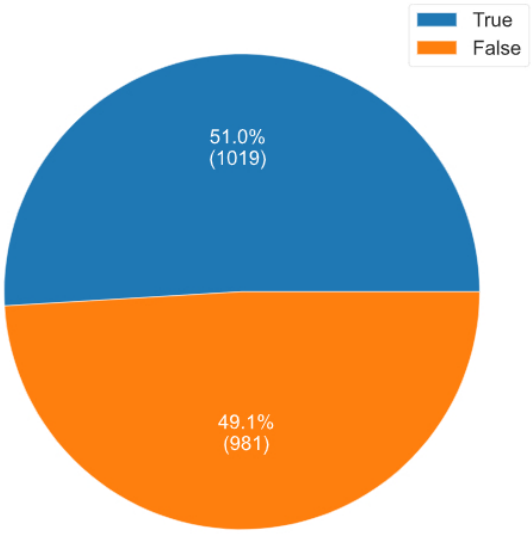
Has dual sim support or not

Distinct	2
Distinct (%)	0.1%
Missing	0
Missing (%)	0.0%
Memory size	2.1 KiB

False 981

Toggle details

Common Values | Chart



~~front_cam_resolution...~~

~~Unsupported~~

REJECTED
UNSUPPORTED

Missing	0
Missing (%)	0.0%
Memory size	86.5 KiB

has_four_g

Boolean

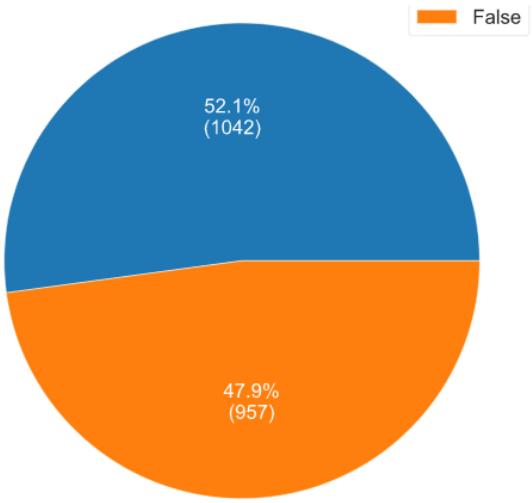
Has 4G or not. 1 = yes , 0 = no

Distinct	2
Distinct (%)	0.1%
Missing	0
Missing (%)	0.0%
Memory size	2.1 KiB

True 1043
False 957

Toggle details

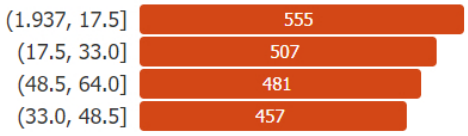
Common Values | Chart



int_memory_category

Categorical

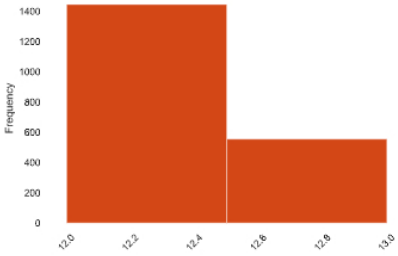
Distinct	4
Distinct (%)	0.2%
Missing	0
Missing (%)	0.0%
Memory size	2.3 KiB



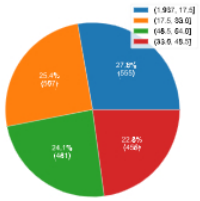
Toggle details

Overview | Categories | Words | Characters

Value	Count	Frequency (%)
(1.937, 1...	555	27.8%
(17.5, 33...	507	25.4%
(48.5, 64...	481	24.1%
(33.0, 48...	457	22.9%



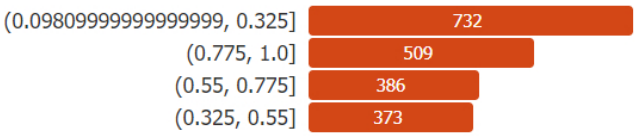
Histogram of lengths of the category



mobile_depth_categ...

Categorical

Distinct	4
Distinct (%)	0.2%
Missing	0
Missing (%)	0.0%
Memory size	2.3 KiB

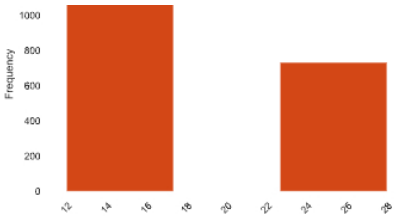


Toggle details

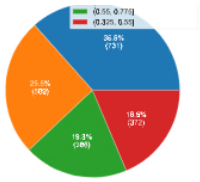
Overview | Categories | Words | Characters

Value	Count	Frequency (%)

(0.09809...	732	36.6%
(0.775, 1...	509	25.4%
(0.55, 0....	386	19.3%
(0.325, 0...	373	18.6%



Histogram of lengths of the category



mobile_weight_categ...

Categorical

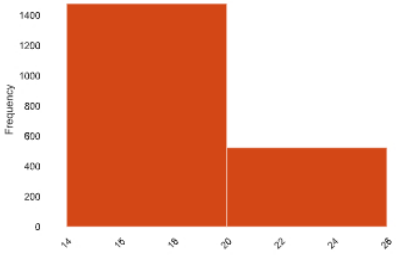
Distinct	4
Distinct (%)	0.2%
Missing	0
Missing (%)	0.0%
Memory size	2.3 KiB

(79.87899999999999, 110.0]	523
(140.0, 170.0]	508
(170.0, 200.0]	496
(110.0, 140.0]	473

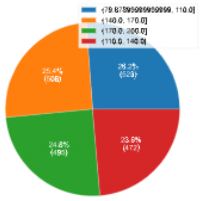
Toggle details

Overview | Categories | Words | Characters

Value	Count	Frequency (%)
(79.8789...	523	26.2%
(140.0, 1...	508	25.4%
(170.0, 2...	496	24.8%
(110.0, 1...	473	23.6%



Histogram of lengths of the category

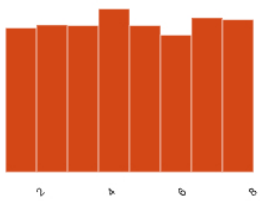


number_of_cores

Real number ($\mathbb{R}_{>0}$)

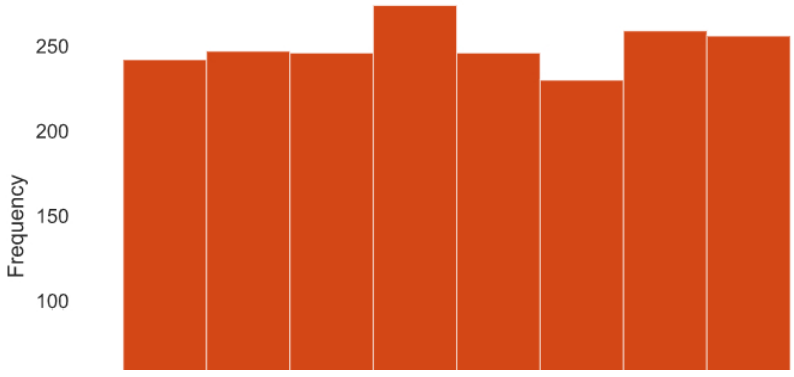
Number of cores of processor

Distinct	8	**Mean**	4.5205
Distinct (%)	0.4%	**Minimum**	1
Missing	0	**Maximum**	8
Missing (%)	0.0%	**Zeros**	0
Infinite	0	**Zeros (%)**	0.0%
Infinite (%)	0.0%	**Memory size**	15.8 KiB



Toggle details

Statistics | Histogram | Common values | Extreme values



Histogram with fixed size bins (bins=8)

~~primary_cam_resolut~~...
~~Unsupported~~

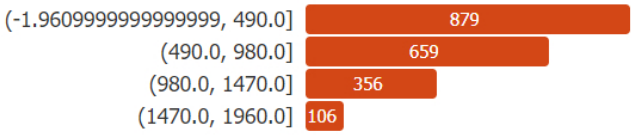
REJECTED
UNSUPPORTED

Missing	0
Missing (%)	0.0%
Memory size	92.3 KiB

px_height_category

Categorical

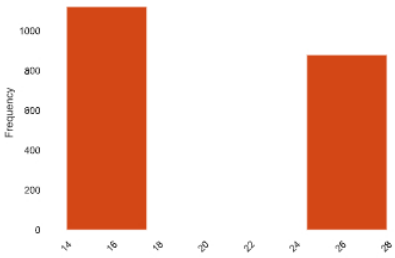
Distinct	4
Distinct (%)	0.2%
Missing	0
Missing (%)	0.0%
Memory size	2.3 KiB



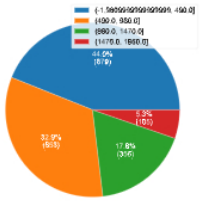
Toggle details

Overview | Categories | Words | Characters

Value	Count	Frequency (%)
(-1.9609...	879	44.0%
(490.0, 9...	659	33.0%
(980.0, 1...	356	17.8%
(1470.0, ...	106	5.3%



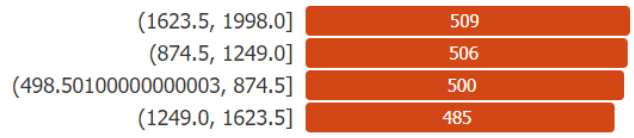
Histogram of lengths of the category



px_width_category

Categorical

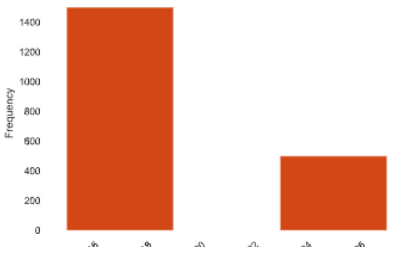
Distinct	4
Distinct (%)	0.2%
Missing	0
Missing (%)	0.0%
Memory size	2.3 KiB

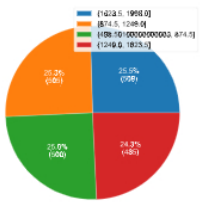


Toggle details

Overview | Categories | Words | Characters

Value	Count	Frequency (%)
(1623.5, ...	509	25.4%
(874.5, 1...	506	25.3%
(498.501...	500	25.0%
(1249.0, ...	485	24.2%





Histogram of lengths of the category

ram_category

Categorical

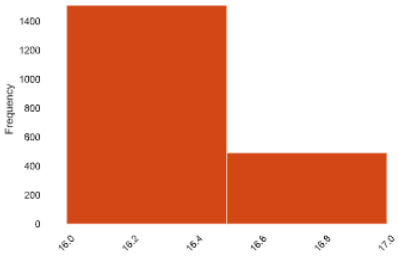
Distinct	4
Distinct (%)	0.2%
Missing	0
Missing (%)	0.0%
Memory size	2.3 KiB

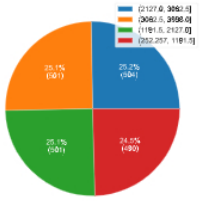
(2127.0, 3062.5] 505
(3062.5, 3998.0] 502
(1191.5, 2127.0] 502
(252.257, 1191.5] 491

Toggle details

Overview | Categories | Words | Characters

Value	Count	Frequency (%)
(2127.0, ...	505	25.2%
(3062.5, ...	502	25.1%
(1191.5, ...	502	25.1%
(252.257...	491	24.6%





Histogram of lengths of the category

screen_height_categ...

Categorical

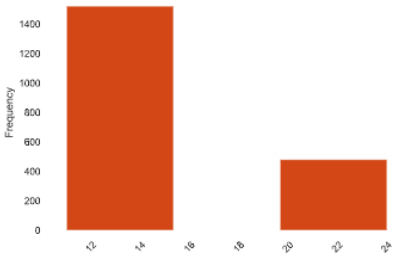
Distinct	4
Distinct (%)	0.2%
Missing	0
Missing (%)	0.0%
Memory size	2.3 KiB

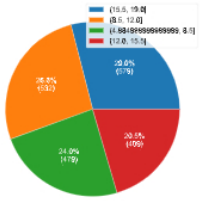
(15.5, 19.0] 580
(8.5, 12.0] 532
(4.984999999999999, 8.5] 479
(12.0, 15.5] 409

Toggle details

Overview | Categories | Words | Characters

Value	Count	Frequency (%)
(15.5, 19...	580	29.0%
(8.5, 12.0]	532	26.6%
(4.98499...	479	23.9%
(12.0, 15...	409	20.4%





Histogram of lengths of the category

screen_width_category

Categorical

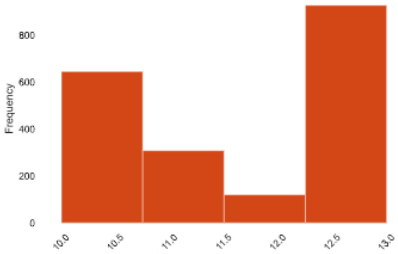
Distinct	4
Distinct (%)	0.2%
Missing	0
Missing (%)	0.0%
Memory size	2.3 KiB

(-0.019, 4.5] 927
(4.5, 9.0] 645
(9.0, 13.5] 308
(13.5, 18.0] 120

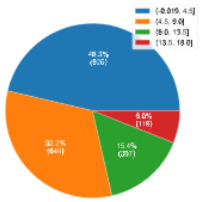
Toggle details

Overview | Categories | Words | Characters

Value	Count	Frequency (%)
(-0.019, ...	927	46.4%
(4.5, 9.0]	645	32.2%
(9.0, 13.5]	308	15.4%
(13.5, 18...	120	6.0%



Histogram of lengths of the category



talk_time_category

Categorical

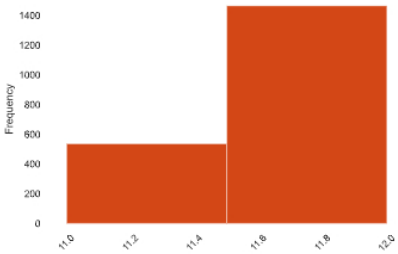
Distinct	4
Distinct (%)	0.2%
Missing	0
Missing (%)	0.0%
Memory size	2.3 KiB

(6.5, 11.0] 536
(15.5, 20.0] 529
(1.981, 6.5] 520
(11.0, 15.5] 415

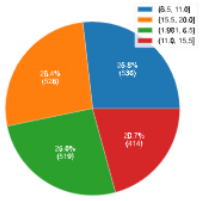
Toggle details

Overview | Categories | Words | Characters

Value	Count	Frequency (%)
(6.5, 11.0]	536	26.8%
(15.5, 20...	529	26.5%
(1.981, 6...	520	26.0%
(11.0, 15...	415	20.8%



Histogram of lengths of the category



has_three_g

Boolean

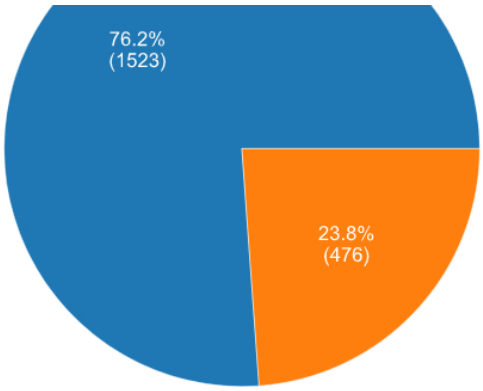
Has 3G or not

Distinct	2
Distinct (%)	0.1%
Missing	0
Missing (%)	0.0%
Memory size	2.1 KiB

True 1523
False 477

Toggle details

Common Values | Chart



has_touch_screen

Boolean

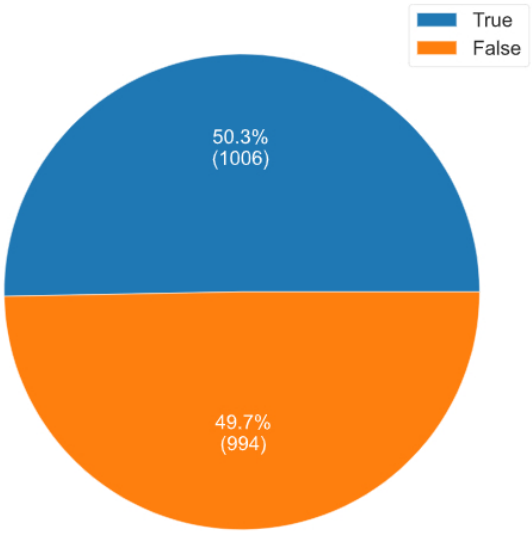
Has touch screen or not, 1 = yes, 0 = no

Distinct	2
Distinct (%)	0.1%
Missing	0
Missing (%)	0.0%
Memory size	2.1 KiB

True	1006
False	994

Toggle details

Common Values | Chart



has_wifi

Boolean

Distinct	2

True	1014
False	986

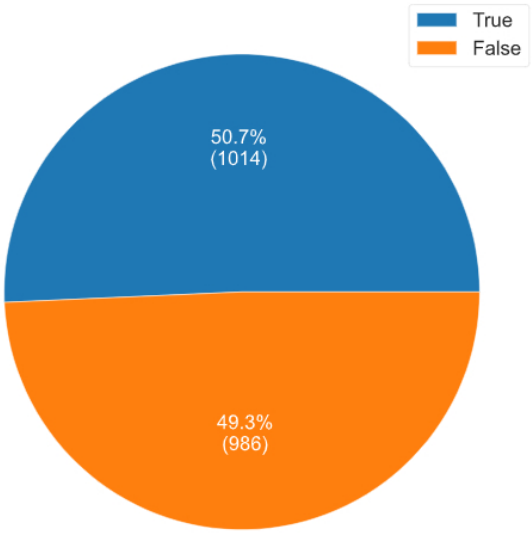
Boolean

Has wifi or not

Distinct (%)	0.1%
Missing	0
Missing (%)	0.0%
Memory size	2.1 KiB

Toggle details

Common Values | Chart



is_expensive

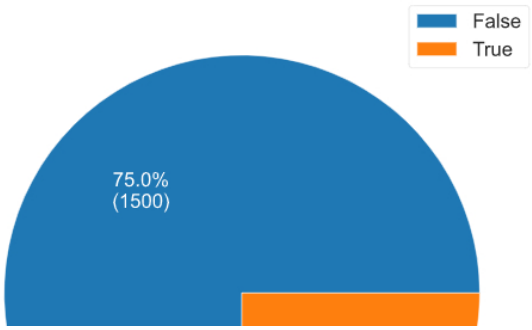
Boolean

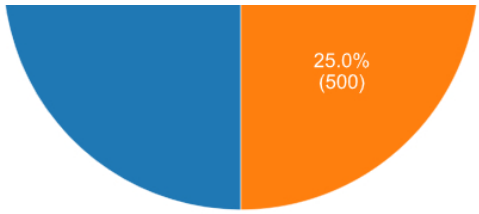
This is the target variable with indicating if the mobile phone got a high price. 1 = yes, 0 = no

Distinct	2
Distinct (%)	0.1%
Missing	0
Missing (%)	0.0%
Memory size	2.1 KiB

Toggle details

Common Values | Chart





Interactions

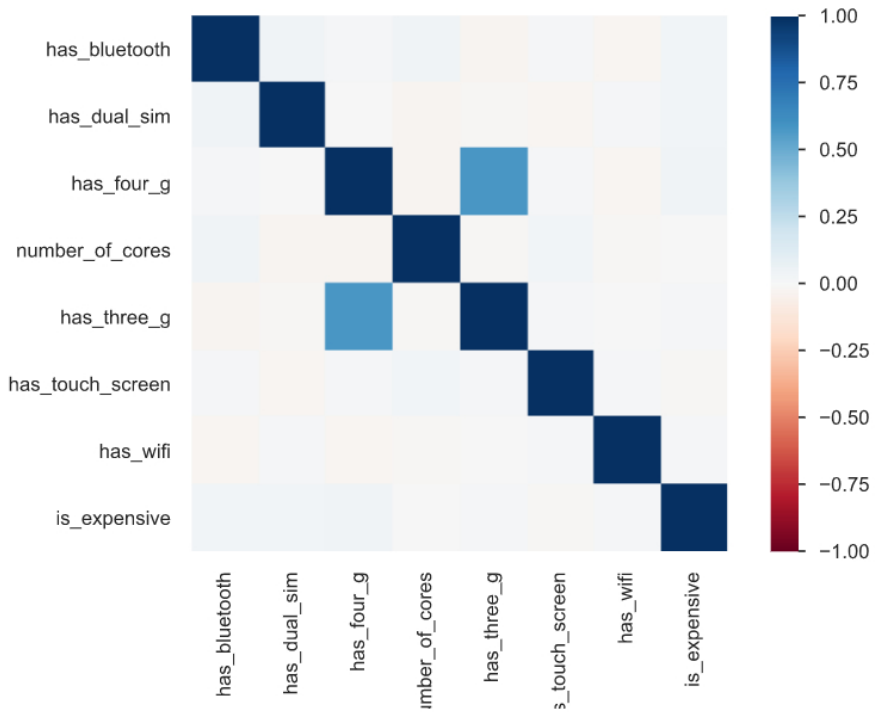
number_of_cores

number_of_cores

Correlations

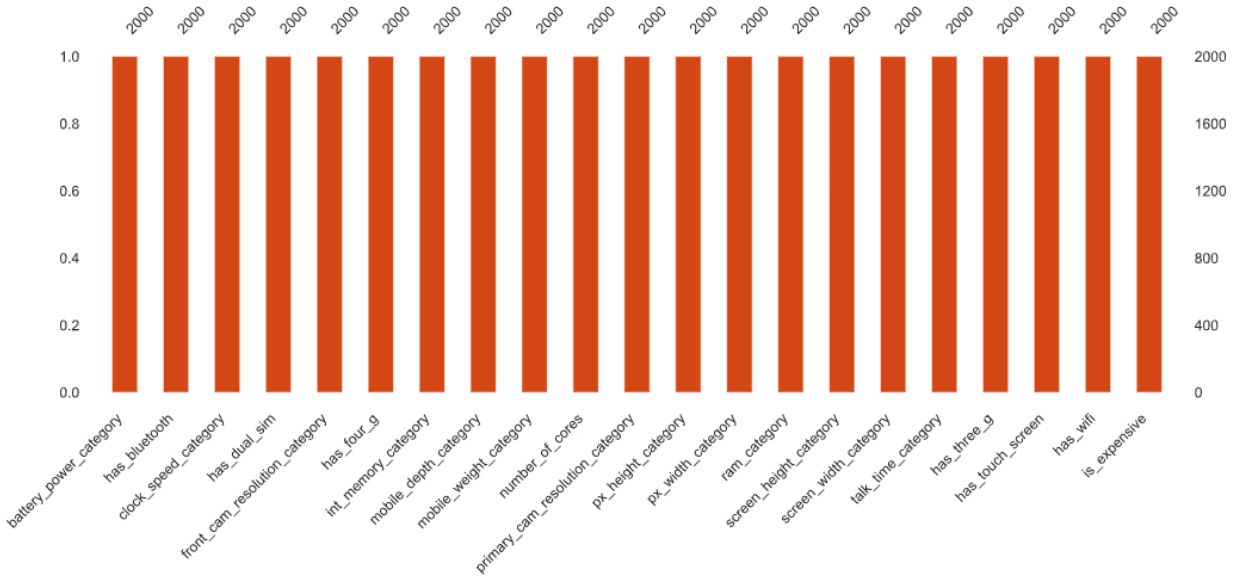
Pearson's r | Spearman's ρ | Kendall's τ | Phik (φk) | Cramér's V (φc)

Toggle correlation descriptions



Missing values

Count | Matrix



A simple visualization of nullity by column.

Sample

First rows

	battery_power_category	has_bluetooth	clock_speed_category	has_dual_sim	front_cam_resolution_category	has_f
0	(499.502, 875.25]	False	(1.75, 2.375]	False	(0.981, 5.5]	False
1	(875.25, 1249.5]	True	(0.497, 1.125]	True	0	True
2	(499.502, 875.25]	True	(0.497, 1.125]	True	(0.981, 5.5]	True
3	(499.502, 875.25]	True	(2.375, 3.0]	False	0	False
4	(1623.75, 1998.0]	True	(1.125, 1.75]	False	(10.0, 14.5]	True
5	(1623.75, 1998.0]	False	(0.497, 1.125]	True	(0.981, 5.5]	False
6	(1623.75, 1998.0]	False	(1.125, 1.75]	False	(0.981, 5.5]	True
7	(1623.75, 1998.0]	False	(0.497, 1.125]	True	0	False
8	(1249.5, 1623.75]	True	(0.497, 1.125]	False	0	False
9	(499.502, 875.25]	True	(0.497, 1.125]	True	(0.981, 5.5]	True

Last rows

	battery_power_category	has_bluetooth	clock_speed_category	has_dual_sim	front_cam_resolution_category	ha
1990	(1249.5, 1623.75]	True	(2.375, 3.0]	False	(5.5, 10.0]	Tr
1991	(1623.75, 1998.0]	False	(1.75, 2.375]	False	(10.0, 14.5]	Tr
1992	(499.502, 875.25]	True	(2.375, 3.0]	True	(0.981, 5.5]	Fa
1993	(1249.5, 1623.75]	True	(0.497, 1.125]	False	0	Fa
1994	(499.502, 875.25]	False	(1.75, 2.375]	False	(0.981, 5.5]	Fa
1995	(499.502, 875.25]	True	(0.497, 1.125]	True	0	Tr
1996	(1623.75, 1998.0]	True	(2.375, 3.0]	True	0	Fa
1997	(1623.75, 1998.0]	False	(0.497, 1.125]	True	(0.981, 5.5]	Tr
1998	(1249.5, 1623.75]	False	(0.497, 1.125]	False	(0.981, 5.5]	Tr
1999	(499.502, 875.25]	True	(1.75, 2.375]	True	(0.981, 5.5]	Tr