

LAPORAN
***FINAL PROJECT* KECERDASAN KOMPUTASIONAL**




Mobile Price Classification

Dipersiapkan oleh:

Christian Bennett R
Christoffer Ivano
Melanchthon Bonifacio B

05111940000078
05111940000091
05111940000097

Jurusan Teknik Informatika - Institut Teknologi Sepuluh Nopember
Kampus ITS Keputih Sukolilo Surabaya

	Jurusan Teknik Informatika ITS	Dosen Pengampu		Halaman
		<i>Prof.Ir. Handayani Tjandrasa, M.Sc Ph.D</i>		
		Revisi	-	<i>30 November 2021</i>

Daftar Isi

• <i>Cover</i>	i
• Daftar Isi	ii
• 1.1 Latar Belakang	3
• 1.2 Tujuan dan Manfaat	3
• 2.1 Deskripsi Penelitian	4
• 3.1 Implementasi	5
• 3.1.1 <i>Decision Tree</i>	5
• 3.1.2 <i>K-Nearest Neighbor</i>	5
• 3.1.3 <i>Random Forest</i>	5
• 3.1.4 <i>Support Vector Machine</i>	6

BAB 1

1.1 Latar Belakang

Alat komunikasi pada saat ini merupakan suatu kebutuhan yang sangat penting bagi masyarakat. Pada zaman modern seperti sekarang ini banyak sekali jenis alat komunikasi, misalnya telepon rumah, handphone atau bisa melalui internet (chatting, e-mail). Dengan majunya ilmu teknologi pada saat ini handphone tidak hanya dapat digunakan untuk telepon dan mengirim pesan singkat, namun dapat digunakan untuk chatting, membuka email, ataupun komunikasi lainnya melalui internet. Dengan bentuknya yang kecil, mudah dibawa dan memiliki aplikasi yang lengkap maka handphone merupakan salah satu pilihan yang tepat.

Pada saat ini permintaan masyarakat terhadap handphone semakin meningkat, hal ini merupakan peluang bagi produsen handphone untuk menciptakan inovasi-inovasi baru dalam pembuatan handphone yang dihasilkannya. Salah satu produsen handphone yang disegani saat ini adalah samsung, pasalnya perusahaan asal korea selatan tersebut berhasil menguasai pangsa pasar handphone dunia. Bahkan Samsung telah menciptakan lima puluh persen tren baru, sebagai contoh adalah smartphone layar lebar yang dirilis Samsung pada tahun 2011 lalu. Analisis dari strategi analytics, mawston (2011) mengatakan pertumbuhan Samsung yang mengesankan tersebut disebabkan oleh desain ponsel yang menarik, fitur yang canggih dan penggunaan sistem android serta jaringan distribusi yang luas secara global. Sehingga apa yang dilakukan Samsung ditiru oleh produsen handphone terkemuka.

Untuk mengatasi masalah ini, dikumpulkannya data penjualan ponsel dari berbagai perusahaan. Agar menghasilkan *handphone* yang sesuai dengan harga dan spesifikasi yang disediakan maka kami menggunakan beberapa metode *Machine Learning*, yaitu *K-nearest Neighbor* (KNN), *Support Vector Machine* (SVM), *Decision Tree*, dan *Random Forest*. Untuk pre-processing menggunakan *missing value handle*, *feature selection*, dan *outliers data checking*.

1.2 Tujuan dan Manfaat

Tujuan dari penelitian ini adalah menentukan dan mendapatkan range harga terbaik untuk sebuah *handphone* dengan spesifikasi tertentu. Selain itu, juga menemukan metode klasifikasi dengan akurasi yang terbaik dari yang lainnya.

BAB 2

2.1 Deskripsi Penelitian

Untuk atribut *dataset* Mobile Price Classification ini adalah sebagai berikut:

Atribut	Deskripsi
battery_power	Total energi yang dapat disimpan dalam baterai dalam satu siklus pengisian daya (dalam mAh)
blue	Ketersediaan bluetooth
clock_speed	Kecepatan microprocessor menjalankan instruksi
dual_sim	Ketersediaan dukungan untuk dual sim
fc	Besarnya megapixel kamera depan
four_g	Ketersediaan 4G
int_memory	Besarnya memori internal (dalam GB)
m_dep	Lebar handphone (dalam cm)
mobile_wt	Berat handphone (dalam g)
n_cores	Jumlah core processor
pc	Besar megapixel kamera utama
px_height	Tinggi resolusi pixel
px_width	Lebar resolusi pixel
ram	Besarnya RAM (dalam MB)
sc_h	Tinggi layar (dalam cm)
sc_w	Lebar layar (dalam cm)
talk_time	Waktu terpanjang yang bisa digunakan untuk menelpon dalam satu siklus pengisian daya
three_g	Ketersediaan 3G
touch_screen	Ketersediaan layar sentuh
wifi	Ketersediaan wifi
price_range(Class)	Jangkauan harga dengan nilai 0 (biaya rendah) - 3 (biaya tinggi)

BAB 3

3.1 Implementasi Kecerdasan Komputasional pada FP

Untuk mendapatkan data harga yang terbaik kami menggunakan beberapa metode *Machine Learning*, yaitu *K-nearest Neighbor* (KNN), *Support Vector Machine* (SVM), *Decision Tree*, dan *Random Forest*.

3.1.1 Decision Tree

Decision tree adalah salah satu metode klasifikasi yang paling populer, karena mudah untuk diinterpretasi oleh manusia. *Decision tree* adalah model prediksi menggunakan struktur pohon atau struktur berhirarki. Manfaat utama dari penggunaan *decision tree* adalah kemampuannya untuk mem-*break down* proses pengambilan keputusan yang kompleks menjadi lebih simple, sehingga pengambil keputusan akan lebih menginterpretasikan solusi dari permasalahan. Kelebihan lain dari metode ini adalah mampu mengeliminasi perhitungan atau data-data yang kiranya tidak diperlukan. Sebab, sampel yang ada biasanya hanya diuji berdasarkan kriteria atau kelas tertentu saja.

3.1.2 K-Nearest Neighbor

Algoritma *k-Nearest Neighbor* adalah algoritma *supervised learning* dimana hasil dari instance yang baru diklasifikasikan berdasarkan mayoritas dari kategori k-tetangga terdekat. Tujuan dari algoritma ini adalah untuk mengklasifikasikan objek baru berdasarkan atribut dan sampel-sample dari *training data*. Algoritma *k-Nearest Neighbor* menggunakan *Neighborhood Classification* sebagai nilai prediksi dari nilai *instance* yang baru.

Kita bisa menentukannya dengan menggunakan Algoritma k-NN, yaitu dengan melibatkan jarak antara rumah tersebut dengan rumah-rumah yang ada disekitarnya (tetangganya). Pertama, kita harus menentukan jumlah tetangga yg akan kita perhitungkan (k), misalnya kita tentukan 3 tetangga terdekat (k = 3). Kedua, hitung jarak setiap tetangga terhadap rumah tersebut, lalu urutkan hasilnya berdasarkan jarak, mulai dari yang terkecil ke yang terbesar. Ketiga, ambil 3 (k) tetangga yg paling dekat, lalu kita lihat masing-masing dari tetangga tersebut apakah termasuk kedalam wilayah Kota atau Kabupaten.

3.1.3 Random Forest

Random forest (RF) adalah suatu algoritma yang digunakan pada klasifikasi data dalam jumlah yang besar. Klasifikasi *random forest* dilakukan melalui penggabungan pohon (*tree*) dengan melakukan *training* pada sampel data yang dimiliki. Penggunaan pohon (*tree*) yang semakin banyak akan mempengaruhi akurasi yang akan didapatkan menjadi lebih baik. Penentuan klasifikasi dengan *random forest* diambil berdasarkan hasil *voting* dari *tree* yang terbentuk. Pemenang dari *tree* yang terbentuk ditentukan dengan *vote* terbanyak. Pembangunan pohon (*tree*) pada *random forest* sampai dengan mencapai ukuran maksimum dari pohon data. Akan tetapi, pembangunan pohon *random forest* tidak dilakukan pemangkasan (*pruning*) yang merupakan sebuah metode untuk mengurangi kompleksitas ruang. Pembangunan dilakukan dengan penerapan metode *random feature selection* untuk meminimalisir kesalahan. Pembentukan pohon (*tree*) dengan sampel data menggunakan variabel yang diambil secara acak dan menjalankan klasifikasi pada semua *tree* yang terbentuk.

Random forest menggunakan *Decision Tree* untuk melakukan proses seleksi. Pohon yang dibangun dibagi secara rekursif dari data pada kelas yang sama. Pemecahan (*split*) digunakan untuk membagi data berdasarkan jenis atribut yang digunakan. Proses klasifikasi pada *random forest* berawal dari memecah data sampel yang ada ke dalam *decision tree* secara acak. Setelah pohon terbentuk, maka akan dilakukan *voting* pada setiap kelas dari data sampel. Kemudian, mengkombinasikan *vote* dari setiap kelas kemudian diambil *vote* yang paling banyak. Dengan menggunakan *random forest* pada klasifikasi data maka, akan menghasilkan *vote* yang paling baik.

3.1.4 Support Vector Machine

Support Vector Machine (SVM) merupakan salah satu metode dalam *supervised learning* yang biasanya digunakan untuk klasifikasi (seperti *Support Vector Classification*) dan regresi (*Support Vector Regression*). Dalam pemodelan klasifikasi, SVM memiliki konsep yang lebih matang dan lebih jelas secara matematis dibandingkan dengan teknik-teknik klasifikasi lainnya. SVM juga dapat mengatasi masalah klasifikasi dan regresi dengan *linear* maupun *non linear*.

SVM digunakan untuk mencari *hyperplane* terbaik dengan memaksimalkan jarak antar kelas. *Hyperplane* adalah sebuah fungsi yang dapat digunakan untuk pemisah antar kelas. Dalam 2-D fungsi yang digunakan untuk klasifikasi antar kelas disebut sebagai *line* whereas, fungsi yang digunakan untuk klasifikasi antar kelas dalam 3D disebut *plane* similarly, sedangkan fungsi yang digunakan untuk klasifikasi di dalam ruang kelas dimensi yang lebih tinggi disebut *hyperplane*.

BAB 4

4.1 Import Library dan Dataset

```
In [1]: import numpy as np # Linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
```

```
In [2]: # Data Visualization
import matplotlib.pyplot as plt
import seaborn as sns
# Train/Test Split
from sklearn.model_selection import train_test_split
```

```
In [3]: df = pd.read_csv('C:/Users/Christoffer Ivano/Downloads/trainFP.csv')
df.head()
```

```
Out[3]:
```

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

5 rows × 21 columns

```
In [4]: df.info() #tipe data
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2000 entries, 0 to 1999
Data columns (total 21 columns):
#   Column                Non-Null Count  Dtype
---  -
0   battery_power         2000 non-null  int64
1   blue                  2000 non-null  int64
2   clock_speed           2000 non-null  float64
3   dual_sim              2000 non-null  int64
4   fc                    2000 non-null  int64
5   four_g                2000 non-null  int64
6   int_memory            2000 non-null  int64
7   m_dep                 2000 non-null  float64
8   mobile_wt             2000 non-null  int64
9   n_cores               2000 non-null  int64
10  pc                     2000 non-null  int64
11  px_height              2000 non-null  int64
12  px_width              2000 non-null  int64
13  ram                   2000 non-null  int64
14  sc_h                  2000 non-null  int64
15  sc_w                  2000 non-null  int64
16  talk_time             2000 non-null  int64
17  three_g               2000 non-null  int64
18  touch_screen          2000 non-null  int64
19  wifi                  2000 non-null  int64
20  price_range           2000 non-null  int64
dtypes: float64(2), int64(19)
memory usage: 328.2 KB
```

```
In [5]: df.describe() #persebaran
```

```
Out[5]:
```

	battery_power	blue	clock_speed	dual_sim	fc	four_g	int_memory	m_dep	mobile_wt	n_cores	...	px_height
count	2000.000000	2000.0000	2000.000000	2000.000000	2000.000000	2000.000000	2000.000000	2000.000000	2000.000000	2000.000000	...	2000.000000
mean	1238.518500	0.4950	1.522250	0.509500	4.309500	0.521500	32.046500	0.501750	140.249000	4.520500	...	645.108000
std	439.418206	0.5001	0.816004	0.500035	4.341444	0.499662	18.145715	0.288416	35.399655	2.287837	...	443.780811
min	501.000000	0.0000	0.500000	0.000000	0.000000	0.000000	2.000000	0.100000	80.000000	1.000000	...	0.000000
25%	851.750000	0.0000	0.700000	0.000000	1.000000	0.000000	16.000000	0.200000	109.000000	3.000000	...	282.750000
50%	1226.000000	0.0000	1.500000	1.000000	3.000000	1.000000	32.000000	0.500000	141.000000	4.000000	...	564.000000
75%	1615.250000	1.0000	2.200000	1.000000	7.000000	1.000000	48.000000	0.800000	170.000000	7.000000	...	947.250000
max	1998.000000	1.0000	3.000000	1.000000	19.000000	1.000000	64.000000	1.000000	200.000000	8.000000	...	1960.000000

8 rows × 13 columns

4.2 Data Preprocessing

4.2.1 Menangani Missing Values

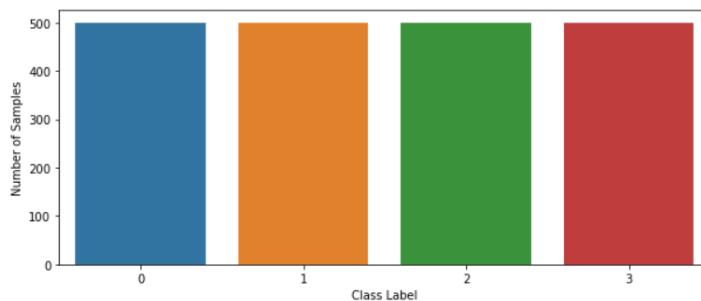
```
In [6]: df.isna().sum() #perncarian null
```

```
Out[6]: battery_power    0
blue                    0
clock_speed             0
dual_sim               0
fc                     0
four_g                 0
int_memory             0
m_dep                 0
mobile_wt              0
n_cores               0
pc                    0
px_height              0
px_width               0
ram                   0
sc_h                  0
sc_w                  0
talk_time              0
three_g               0
touch_screen          0
wifi                  0
price_range            0
dtype: int64
```

Pada gambar di atas dapat dilihat bahwa pada *dataset* yang digunakan, tidak terdapat *missing values*.

4.2.2 Keseimbangan Data

```
In [10]: #cek persebaran price range
fig, ax = plt.subplots(figsize = (10, 4))
sns.countplot(x = 'price_range', data=df)
plt.xlabel("Class Label")
plt.ylabel("Number of Samples")
plt.show()
```

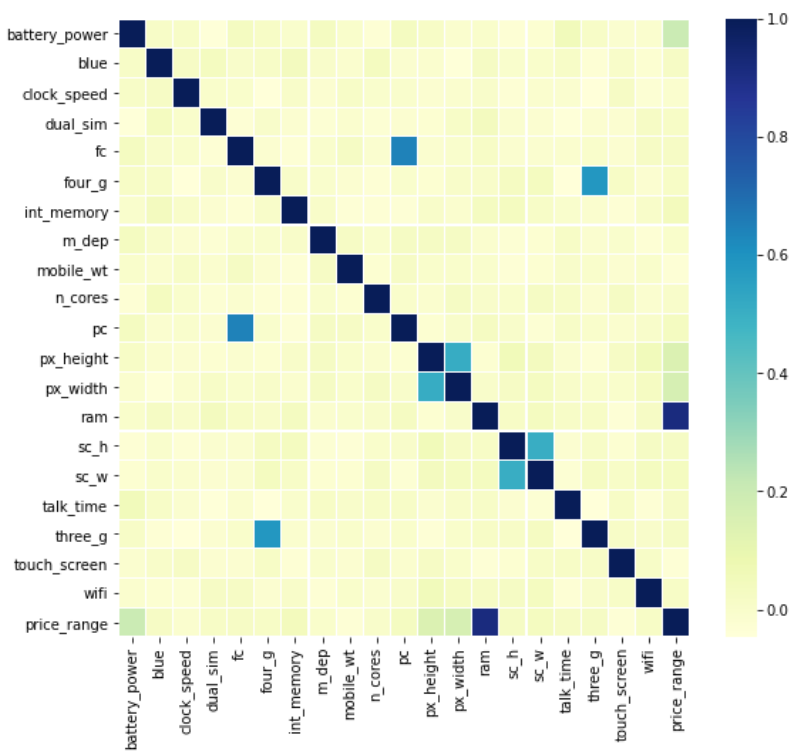


Dapat dilihat bahwa label tersebar secara merata dalam 2000 data yang dimiliki, dimana setiap label memiliki 500 data, sehingga tidak perlu dilakukan *data balancing*.

4.2.3 Feature Selection menggunakan Dataframe Correlation


```
In [12]: corr_mat = df.corr()
f, ax = plt.subplots(figsize = (9, 8))
sns.heatmap(corr_mat, ax = ax, cmap = "YlGnBu", linewidths = 0.1)
```

Out[12]: <AxesSubplot:>



```
In [13]: corr_mat['price_range']
```

```
Out[13]: battery_power    0.200723
blue                    0.020573
clock_speed            -0.006606
dual_sim               0.017444
fc                     0.021998
four_g                 0.014772
int_memory             0.044435
m_dep                  0.000853
mobile_wt              -0.030302
n_cores                0.004399
pc                     0.033599
px_height              0.148858
px_width               0.165818
ram                    0.917046
sc_h                   0.022986
sc_w                   0.038711
talk_time              0.021859
three_g                0.023611
touch_screen           -0.030411
wifi                   0.018785
price_range            1.000000
Name: price_range, dtype: float64
```

```
In [15]: # Absolute
corr_mat['price_range'] = abs(corr_mat['price_range'])
sorted_corr_mat = corr_mat.sort_values(by=['price_range'], ascending=False)
sorted_corr_mat['price_range']
```

```
Out[15]: price_range    1.000000
ram                0.917046
battery_power      0.200723
px_width           0.165818
px_height          0.148858
int_memory         0.044435
sc_w               0.038711
pc                 0.033599
touch_screen       0.030411
mobile_wt          0.030302
three_g            0.023611
sc_h               0.022986
fc                 0.021998
talk_time          0.021859
blue               0.020573
wifi               0.018785
dual_sim           0.017444
four_g             0.014772
clock_speed        0.006606
n_cores            0.004399
m_dep              0.000853
Name: price_range, dtype: float64
```

Setelah melihat *correlation matrix*, korelasi masing-masing fitur tidak direpresentasikan dengan baik, sehingga kami perlu mengeluarkan angka *correlation feature* terhadap 'price_range'.

```
In [16]: # Selecting features with correlation more than 0.022
features = ['ram', 'battery_power', 'px_width', 'px_height', 'int_memory', 'sc_w', 'pc', 'touch_screen', 'mobile_wt', 'three_g',
df = df[features]
```

```
Out[16]:
```

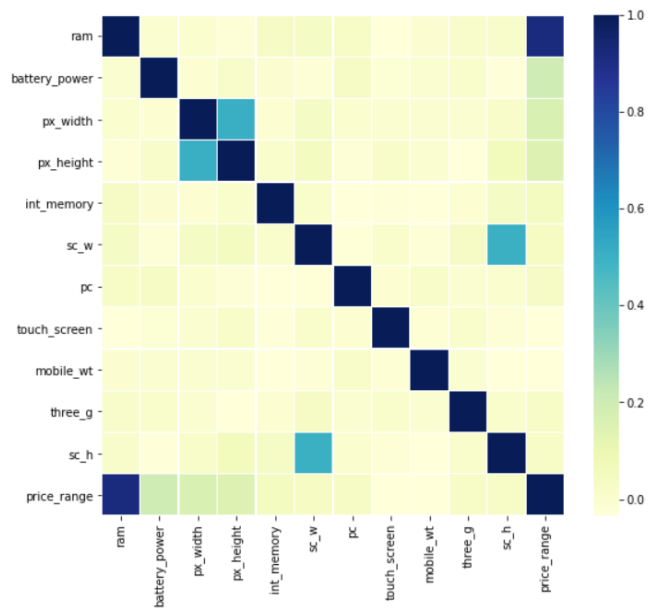
	ram	battery_power	px_width	px_height	int_memory	sc_w	pc	touch_screen	mobile_wt	three_g	sc_h	price_range
0	2549	842	756	20	7	7	2	0	188	0	9	1
1	2631	1021	1988	905	53	3	6	1	136	1	17	2
2	2603	563	1716	1263	41	2	6	1	145	1	11	2
3	2769	615	1786	1216	10	8	9	0	131	1	16	2
4	1411	1821	1212	1208	44	2	14	1	141	1	8	1
...
1995	668	794	1890	1222	2	4	14	1	106	1	13	0
1996	2032	1965	1965	915	39	10	3	1	187	1	11	2
1997	3057	1911	1632	868	36	1	3	1	108	1	9	3
1998	869	1512	670	336	46	10	5	1	145	1	18	0
1999	3919	510	754	483	45	4	16	1	168	1	19	3

2000 rows x 12 columns

Dari angka yang kami dapatkan, kami menganalisis dan menyimpulkan bahwa fitur terbaik adalah 'ram', 'battery_power', 'px_width', 'px_height', 'int_memory', 'sc_w', 'pc', 'touch_screen', 'mobile_wt', 'three_g', dan 'sc_h' karena fitur-fitur tersebut memiliki nilai korelasi dan pengaruh yang lebih tinggi terhadap range harga jika dibandingkan dengan fitur lainnya dalam sebuah *mobile phone*.

```
In [17]: corr_mat = df.corr()
f, ax = plt.subplots(figsize = (9, 8))
sns.heatmap(corr_mat, ax = ax, cmap = "YlGnBu", linewidths = 0.1)
```

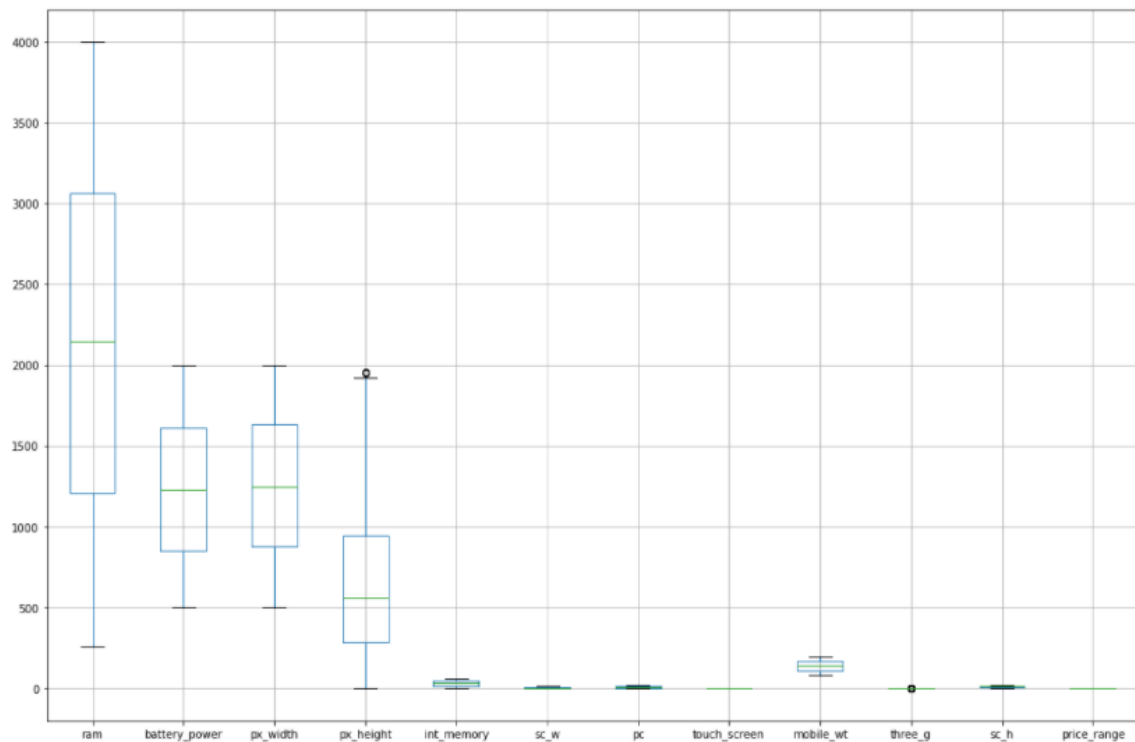
Out[17]: <AxesSubplot:>



4.2.4 Mengecek Data Outlier

```
In [18]: # Check outlier
f, ax = plt.subplots(figsize = (18, 12))
df.boxplot()
```

Out[18]: <AxesSubplot:>



```
In [23]: df.sort_values(by=['px_height'], ascending=False).head(10)
```

```
Out[23]:
```

	ram	battery_power	px_width	px_height	int_memory	sc_w	pc	touch_screen	mobile_wt	three_g	sc_h	price_range
1771	1622	1230	1963	1960	48	17	2	1	111	1	18	2
988	2973	1413	1994	1949	45	8	6	0	104	1	17	3
894	1329	1497	1933	1920	32	1	9	1	92	1	6	2
260	1027	754	1928	1914	59	3	10	0	178	1	6	1
1163	1783	1930	1920	1901	16	3	12	0	188	0	17	2
1827	3779	1992	1904	1899	2	3	17	1	106	1	9	3
1017	2438	1221	1976	1895	28	11	7	1	139	1	13	3
1397	629	717	1981	1878	14	15	6	0	99	1	16	0
1615	3930	1086	1976	1874	24	1	19	1	135	1	17	3
305	955	1348	1942	1869	52	11	20	1	98	1	18	1

Dari gambar diatas, ditemukan 2 atribut yang memiliki *outlier* yaitu pada 'px_height' dan 'three_g', namun setelah diselidiki, data tersebut tidak terlalu mempengaruhi *dataset* dan masih di batas wajar sehingga dibiarkan.

4.3 Splitting Dataset

Disini kami melakukan *splitting dataset* dengan perbandingan *train* terhadap *test* yaitu 8:2 dan memisahkan dan juga membuat fungsi untuk mengeluarkan hasil 'accuracy_score', 'precision_score', 'recall_score', dan 'f1_score'.

```
In [27]: # Separates Label from dataset
```

```
Target = np.array(df.pop('price_range'))
Data = np.array(df)
X_train, X_test, Y_train, Y_test = train_test_split(Data, Target, test_size=0.2, random_state=42, stratify=Target)
```

```
In [31]: # Metric Evaluation
```

```
from sklearn.metrics import make_scorer, accuracy_score, precision_score, recall_score, f1_score
def get_metrics(true_labels, predicted_labels):
    acc = np.round(accuracy_score(true_labels, predicted_labels), 4)
    prec = np.round(precision_score(true_labels, predicted_labels, average='weighted'), 4)
    recall = np.round(recall_score(true_labels, predicted_labels, average='weighted'), 4)
    f1 = np.round(f1_score(true_labels, predicted_labels, average='weighted'), 4)
    return [acc, prec, recall, f1]
```

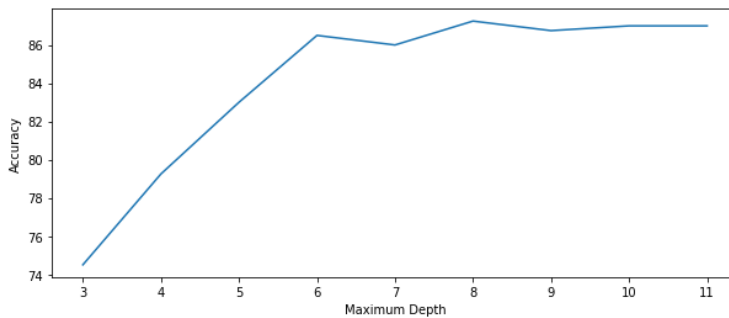
```
In [33]: final_score_headers = ['Model', 'Acc', 'Precision', 'Recall', 'F1']
final_score = pd.DataFrame(columns=final_score_headers)
```

4.4 Decision Tree

```
In [132]: from sklearn. tree import DecisionTreeClassifier
acc = []
max_acc = 0
final_rows = []
x_axis_DT = range(3,12)
for i in x_axis_DT:
    DT = DecisionTreeClassifier(criterion = "gini", random_state = 100, max_depth=i, min_samples_leaf=5)
    DT.fit(X_train, Y_train)
    y_pred = DT.predict(X_test)
    accuracy = accuracy_score(Y_test, y_pred)*100
    if accuracy > max_acc:
        final_rows = ['DT'] + get_metrics(Y_test, y_pred)
        max_acc = accuracy
    acc.append(accuracy)
    print ("Accuracy for Decision Tree for max depth ",i," is: ", accuracy)
print ("Highest accuracy for Decision Tree is: ", max(acc))
final_score = pd.concat([final_score, pd.DataFrame([final_rows], columns=final_score_headers)])
```

```
Accuracy for Decision Tree for max depth 3 is: 74.5
Accuracy for Decision Tree for max depth 4 is: 79.25
Accuracy for Decision Tree for max depth 5 is: 83.0
Accuracy for Decision Tree for max depth 6 is: 86.5
Accuracy for Decision Tree for max depth 7 is: 86.0
Accuracy for Decision Tree for max depth 8 is: 87.25
Accuracy for Decision Tree for max depth 9 is: 86.75
Accuracy for Decision Tree for max depth 10 is: 87.0
Accuracy for Decision Tree for max depth 11 is: 87.0
Highest accuracy for Decision Tree is: 87.25
```

```
In [133]: plt.subplots(figsize = (10, 4))
plt.plot(x_axis_DT, acc)
plt.xlabel('Maximum Depth')
plt.ylabel('Accuracy')
plt.show()
```



4.5 K-Nearest Neighbors

```
In [134]: from sklearn.neighbors import KNeighborsClassifier
acc_KNN = []
max_acc = 0
max_k = 0
final_rows = []
x_axis_KNN = range(1,100)
for i in x_axis_KNN:
    knn = KNeighborsClassifier(n_neighbors=i)
    knn.fit(X_train, Y_train)
    y_pred = knn.predict(X_test)
    accuracy = accuracy_score(Y_test, y_pred)*100
    if accuracy > max_acc:
        final_rows = ['KNN'] + get_metrics(Y_test, y_pred)
        max_acc = accuracy
        max_k = i
    acc_KNN.append(accuracy)
    #print ("Accuracy for KNN for k = ",i," is: ", accuracy)
print ("Highest accuracy for KNN is: ", max(acc_KNN), "at k =", max_k)
final_score = pd.concat([final_score, pd.DataFrame([final_rows], columns=final_score_headers)])
```

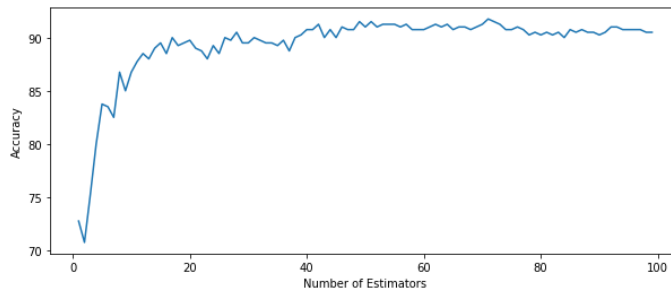
```
Highest accuracy for KNN is: 95.75 at k = 9
```

4.6 Random Forest

```
In [135]: from sklearn.ensemble import RandomForestClassifier
acc_RF = []
max_acc = 0
max_k = 0
final_rows = []
x_axis_RF = range(1,100)
for i in x_axis_RF:
    RF = RandomForestClassifier(n_estimators = i, random_state=0)
    RF.fit(X_train, Y_train)
    y_pred = RF.predict(X_test)
    accuracy = accuracy_score(Y_test, y_pred)*100
    if accuracy > max_acc:
        final_rows = ['RF'] + get_metrics(Y_test, y_pred)
        max_acc = accuracy
        max_k = i
    acc_RF.append(accuracy)
    #print ("Accuracy for Random Forest for estimator = ",i," is: ", accuracy)
print ("Highest accuracy for Random Forest is: ", max(acc_RF), "with", max_k, "estimators.")
final_score = pd.concat([final_score, pd.DataFrame([final_rows], columns=final_score_headers)])
```

Highest accuracy for Random Forest is: 91.75 with 71 estimators.

```
In [136]: plt.subplots(figsize = (10, 4))
plt.plot(x_axis_RF, acc_RF)
plt.xlabel('Number of Estimators')
plt.ylabel('Accuracy')
plt.show()
```



4.7 Support Vector Machine

```
In [185]: from sklearn import svm
#kernel{'linear', 'poly', 'rbf', 'sigmoid', 'precomputed'}
acc_SVM = []
max_acc = 0
final_rows = []
k = ['poly', 'rbf', 'linear', 'sigmoid']
for k in k:
    SVM = svm.SVC(kernel= k)
    SVM.fit(X_train, Y_train)
    y_pred = SVM.predict(X_test)
    accuracy = accuracy_score(Y_test, y_pred)*100
    if accuracy > max_acc:
        final_rows = ['SVM'] + get_metrics(Y_test, y_pred)
        max_acc = accuracy
    acc_SVM.append(accuracy) |
    print ("Accuracy for SVM for kernel = ",k," is: ", accuracy)
print ("Highest accuracy for SVM is: ", max(acc_SVM))
final_score = pd.concat([final_score, pd.DataFrame([final_rows], columns=final_score_headers)])
```

Accuracy for SVM for kernel = poly is: 96.75
 Accuracy for SVM for kernel = rbf is: 96.5
 Accuracy for SVM for kernel = linear is: 98.0
 Accuracy for SVM for kernel = sigmoid is: 12.75
 Highest accuracy for SVM is: 98.0

4.8 Hasil

```
In [186]: final_score = final_score.sort_values(by=['Acc'], ascending=False)
final_score.index = np.arange(1, len(final_score) + 1)
print(final_score)
```

	Model	Acc	Precision	Recall	F1
1	SVM	0.9800	0.9802	0.9800	0.9800
2	KNN	0.9575	0.9573	0.9575	0.9573
3	RF	0.9175	0.9193	0.9175	0.9179
4	DT	0.8725	0.8762	0.8725	0.8732

BAB 5

5.1 Kesimpulan

Sebelum melakukan *feature selection*, *dataset* di cek terlebih dahulu untuk mencari *missing value*. Setelah di cek, tidak terdapat kolom yang mengandung *missing value* satupun.

Pada *feature selection*, terdapat beberapa fitur yang digunakan, yaitu:

1. *RAM*
2. *Battery Power*
3. *Pixel Resolution Width*
4. *Pixel Resolution Height*
5. *Internal Memory*
6. *Phone Screen Height*
7. *Phone Screen Width*
8. *Primary Camera Megapixels*
9. *Touch Screen Support*
10. *Mobile Phone Weight*
11. *3G Support*

Setelah melakukan *feature selection*, di cek apakah terdapat *outlier* atau tidak. Ditemukan bahwa dari *dataset* terdapat 2 atribut yang memiliki *outlier*, yaitu pada atribut 'px_height' dan 'three_g', namun masih di batas wajar sehingga tetap dibiarkan.

Hasil akurasi terbaik yang didapatkan yaitu 98.0% menggunakan metode SVM dengan menggunakan *kernel Linear*.

BAB 6

6.1 Referensi

<https://www.kaggle.com/iabhishekofficial/mobile-price-classification>

<https://towardsdatascience.com/data-preprocessing-e2b0bed4c7fb>

<https://towardsdatascience.com/better-heatmaps-and-correlation-matrix-plots-in-python-41445d0f2bec>

<https://www.geeksforgeeks.org/finding-the-outlier-points-from-matplotlib/>

<https://scikit-learn.org/stable/modules/tree.html>

<https://towardsdatascience.com/the-complete-guide-to-decision-trees-17a874301448>

<https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html>

<https://towardsdatascience.com/knn-k-nearest-neighbors-1-a4707b24bd1d>

<https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html>

<https://towardsdatascience.com/random-forest-3a55c3aca46d>

<https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html>

<https://towardsdatascience.com/support-vector-machine-introduction-to-machine-learning-algorithms-934a444fca47>