FINAL PROJECT – DSC 2021

Implementation Machine Learning by Python

(Car Evaluation Dataset in UCI Machine Learning)

Oleh :

202409001 – Moch Amin Irwanto

PROGRAM STUDI STATISTIKA

FAKULTAS SAINS DAN TEKNOLOGI

UNIVERSITAS PGRI ADI BUANA

SURABAYA

2021

Deskripsi Dataset

1. Judul: Basis Data Evaluasi Mobil
2. Sumber:
   a) Pencipta : Marko Bohanec
   b) Donatur : Marko Bohanec (marko.bohanec@ijs.si)
      Blaz Zupan (blaz.zupan@ijs.si)
   c) Tanggal : June, 1997
3. Penggunaan Sebelumnya:

Model keputusan hierarkis, dari mana kumpulan data ini berasal diturunkan, pertama kali disajikan oleh.

- M. Bohanec dan V. Rajkovic: Perolehan pengetahuan dan penjelasan untuk pengambilan keputusan multi-atribut. Dalam Lokakarya Internasional ke-8 tentang Pakar Sistem dan Aplikasinya, Avignon, Prancis. halaman 59-78, 1988.
- Dalam *Machine Learning*, kumpulan data ini digunakan untuk evaluasi HINT (Hierarchy INduction Tool), yang terbukti mampu sepenuhnya merekonstruksi model hierarkis asli.
- B. Zupan, M. Bohanec, I. Bratko, J. Demsar: Pembelajaran mesin oleh dekomposisi fungsi. ICML-97, Nashville, TN. 1997

4. Informasi yang Relevan:

Basis Data Evaluasi Mobil berasal dari hierarki sederhana model keputusan awalnya dikembangkan untuk demonstrasi DEX (M. Bohanec, V. Rajkovic: Sistem pakar untuk keputusan membuat. Sistemika 1(1), hlm. 145-157, 1990.). Model mengevaluasi mobil sesuai dengan struktur konsep berikut:

Variabel		Deskripsi
Car Price	Buying	harga beli
	Maint	harga pemeliharaan
Tech	Doors	jumlah pintu
	Persons	kapasitas dalam hal orang untuk membawa
	Lug_boot	ukuran bagasi
Safety		perkiraan keamanan mobil
Class		kelas keputusan

Atribut input dicetak dalam huruf kecil. Selain konsep target (CAR), model tersebut mencakup tiga konsep antara: HARGA, TEKNOLOGI, KENYAMANAN. Setiap konsep ada dalam

model aslinya terkait dengan keturunan tingkat yang lebih rendah dengan satu set contoh (untuk kumpulan contoh ini lihat http://www-ai.ijs.si/BlazZupan/car.html).

Basis Data Evaluasi Mobil berisi contoh dengan struktur informasi dihapus, yaitu, secara langsung menghubungkan CAR dengan enam input atribut: pembelian, perawatan, pintu, orang, lug_boot, keamanan. Karena struktur konsep dasar yang diketahui, database ini mungkin sangat berguna untuk menguji induksi konstruktif dan metode penemuan struktur.

5. Jumlah Data: 1727
6. Jumlah Variabel: 6
7. Kategori Variabel:

   buying v-high, high, med, low

   maint v-high, high, med, low

   doors 2, 3, 4, 5-more

   persons 2, 4, more

   lug_boot small, med, big

   safety low, med, high

8. Missing Attribute Values: none
9. Class Distribution (number of instances per class)

class	N	N[%]
unacc	1210	70,02%
acc	384	22,22%
good	69	3,99%
v-good	65	3,76%

Data Visualisasi

Data PreProsessing

Data preprosesing yang dilakukan dalam final project ini adalah sebagai berikut.

- Mengganti nama kolom

	buying	maint	doors	persons	lug_boot	safety	class
0	vhigh	vhigh	2	2	small	med	unacc
1	vhigh	vhigh	2	2	small	high	unacc
2	vhigh	vhigh	2	2	med	low	unacc
3	vhigh	vhigh	2	2	med	med	unacc
4	vhigh	vhigh	2	2	med	high	unacc

- Memeriksa Data Missing Value

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1727 entries, 0 to 1726
Data columns (total 7 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   buying    1727 non-null   object
 1   maint     1727 non-null   object
 2   doors     1727 non-null   object
 3   persons   1727 non-null   object
 4   lug_boot  1727 non-null   object
 5   safety    1727 non-null   object
 6   class     1727 non-null   object
dtypes: object(7)
memory usage: 94.6+ KB
```

- Memeriksa Distribusi Data dan Inkonsistensi Data

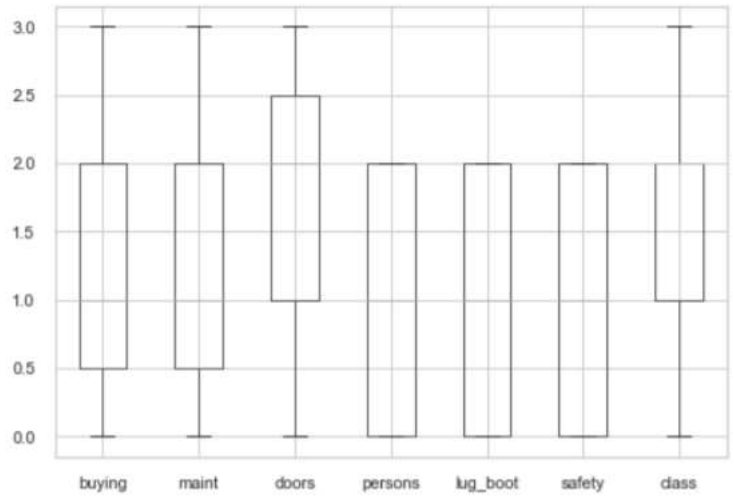
```
med      432
low      432
high     432
vhigh    431
Name: buying, dtype: int64

med      432
low      432
high     432
vhigh    431
Name: maint, dtype: int64

3        432
5more    432
4        432
2        431
Name: doors, dtype: int64

more    576
4       576
2       575
Name: persons, dtype: int64

med      576
big      576
small    575
Name: lug_boot, dtype: int64

med     576
high    576
low     575
Name: safety, dtype: int64

unacc    1209
acc       384
good       69
vgood      65
Name: class, dtype: int64
```

- Categorical Encoding

	buying	maint	doors	persons	lug_boot	safety	class
0	3	vhigh	2	2	small	med	unacc
1	3	vhigh	2	2	small	high	unacc
2	3	vhigh	2	2	med	low	unacc
3	3	vhigh	2	2	med	med	unacc
4	3	vhigh	2	2	med	high	unacc
...	...	...	...	...	...	...	...
1722	1	low	5more	more	med	med	good
1723	1	low	5more	more	med	high	vgood
1724	1	low	5more	more	big	low	unacc
1725	1	low	5more	more	big	med	good
1726	1	low	5more	more	big	high	vgood

1727 rows × 7 columns

- Memeriksa Data Outlier



Regresi Logistik

In [43]:

```
from sklearn.linear_model import LogisticRegression

logreg=LogisticRegression(solver='newton-cg',multi_class='multinomial')
logreg.fit(X_train,y_train)

LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
          intercept_scaling=1, max_iter=100, multi_class='ovr', n_jobs=1,
          penalty='l2', random_state=None, solver='newton-cg', tol=0.0001,
          verbose=0, warm_start=False)

pred=logreg.predict(X_test)
logreg.score(X_test,y_test)
```

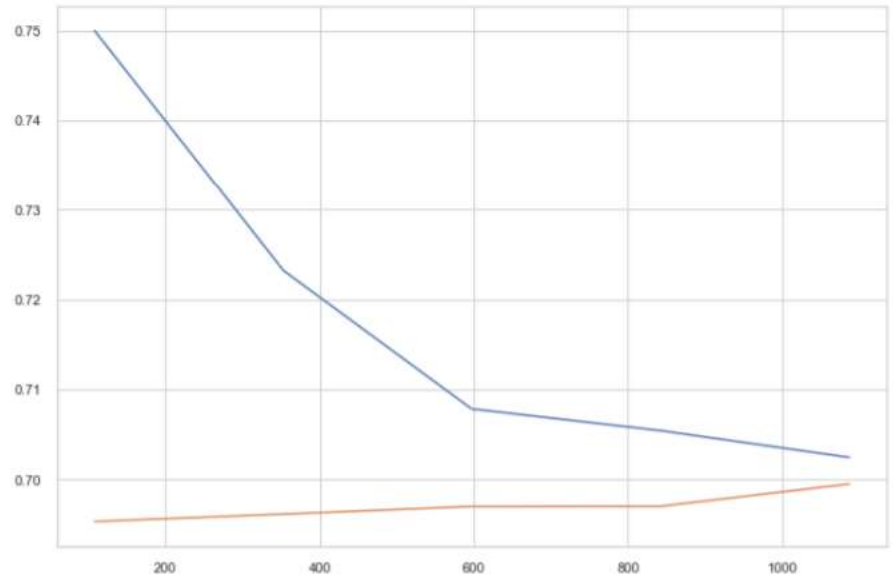
Out[43]:

```
0.6685934489402697
```

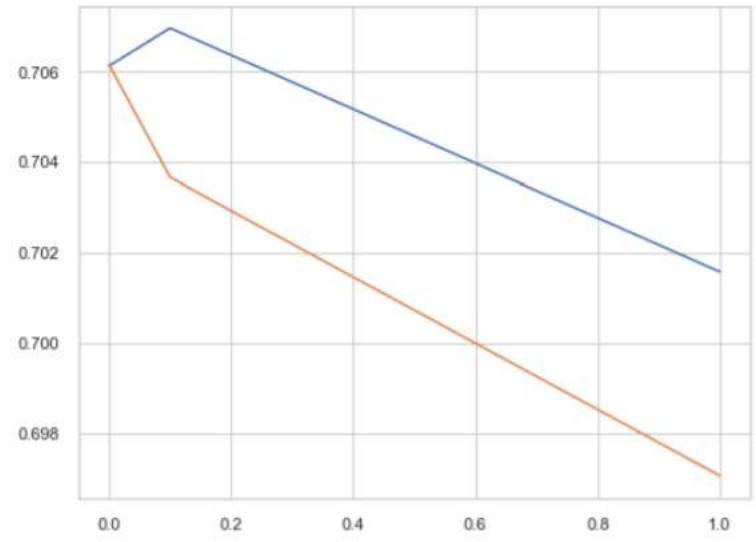
dari logreg.score tersebut diketahui bahwa akurasi yang diperoleh dari model regresi logistik adalah 67%. kemudian kita coba tes modelnya dengan menggunakan learning curve agar dapat dipelajari dan dievaluasi

Out[46]:

```
[<matplotlib.lines.Line2D at 0x1c2e17a1040>]
```



Dari grafik tersebut diketahui bahwa dengan bertambahnya jumlah sampel maka akurasi semakin menurun. disini saya akan mengevaluasi model untuk mencari akurasi terbaik



dari grafik tersebut diketahui bahwa C=0.1 memberikan hasil yang baik.

kemudian lanjut ketahap selanjutnya, karena ini adalah klasifikasi multikelas dan memiliki kumpulan data yang kecil, kita dapat menggunakan GridSearch untuk mendapatkan parameter terbaik.

```
from sklearn.model_selection import GridSearchCV

param_grid={'C':[0.01,0.1,1,10],
            'solver':['newton-cg', 'lbfgs', 'sag'],
            'multi_class':['multinomial']}
grid=GridSearchCV(estimator=LogisticRegression(n_jobs=-1),param_grid=param_grid,cv=5,n_jobs

grid.fit(X_train,y_train)
```

Out[51]:

```
GridSearchCV(cv=5, estimator=LogisticRegression(n_jobs=-1), n_jobs=-1,
             param_grid={'C': [0.01, 0.1, 1, 10],
                         'multi_class': ['multinomial'],
                         'solver': ['newton-cg', 'lbfgs', 'sag']})
```

In [52]:

```
print(grid.best_params_)
print(grid.best_score_)
```

```
{'C': 0.01, 'multi_class': 'multinomial', 'solver': 'newton-cg'}
0.7102705668529886
```

Jadi, dengan parameter tersebut, kita bisa mendapatkan akurasi 71%. karena akurasi regresi masih rendah mari kita bandingkan dengan model kasifikasi KNN

KNN Classifier

In [54]:

```
from sklearn.neighbors import KNeighborsClassifier

knn=KNeighborsClassifier(n_jobs=-1)

knn.fit(X_train,y_train)
pred=knn.predict(X_test)
knn.score(X_test,y_test)
```

Out[54]:

```
0.8747591522157996
```

dari knn.score tersebut diketahui bahwa akurasi model yang didapatkan adalah 87%

In [56]:

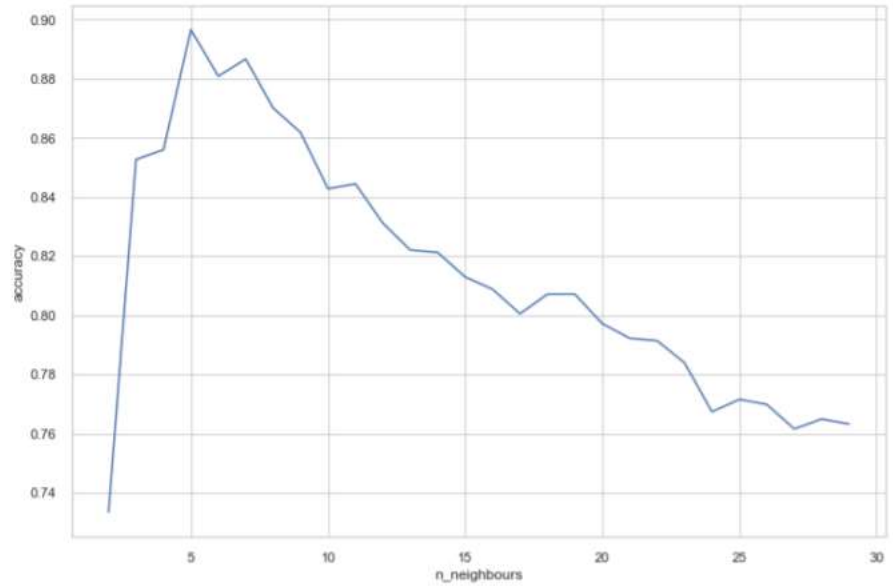
```
from sklearn.metrics import classification_report

print(classification_report(y_test,pred))
```

```
              precision    recall  f1-score   support

           0       0.75      0.74      0.74       115
           1       0.67      0.38      0.48        21
           2       0.92      0.98      0.95       356
           3       1.00      0.48      0.65        27

    accuracy                           0.87       519
   macro avg       0.83      0.64      0.71       519
weighted avg       0.87      0.87      0.87       519
```

untuk memastikan kembali disini saya menggunakan f1-score juga untuk mengevaluasi model klasifikasi. dari hasil f-score tersebut diketahui bahwa akurasi yang didapatkan adalah 87%.

Sekarang saya memeriksa semua parameter untuk 'n_neighbours' dengan menggunakan cross validation sehingga saya dapat mevisualisasikan efek n_neighbours.

```
Text(0, 0.5, 'accuracy')
```



Jadi, dengan algoritma KNN Classification dan n_neighbours=5 memberikan akurasi yang terbaik dengan akurasi sekitar lebih dari 89%. dapat sisimpulkan bahwa model yang dihasilkan dengan menggunakan KNN classifier lebih baik daripada regresi logistik. kemudian saya akan mencoba membandingkan dengan random forest classifier.

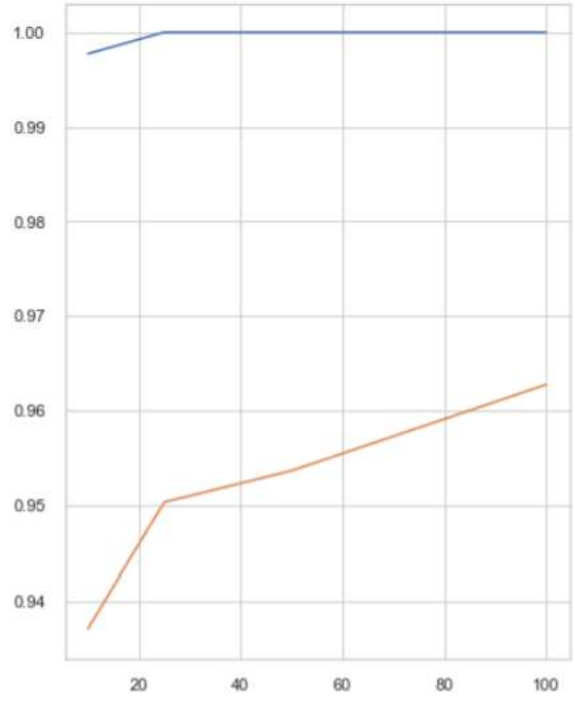
Random Forest Classifier

In [67]:

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import f1_score


rfc=RandomForestClassifier(n_jobs=-1,random_state=51)

rfc.fit(X_train,y_train)
print(rfc.score(X_test,y_test))
print(f1_score(y_test,rfc.predict(X_test),average="macro"))
```

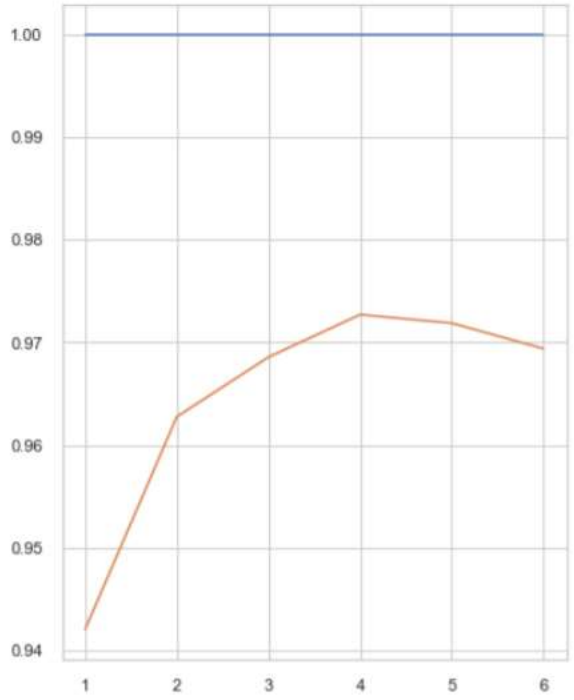
```
0.953757225433526
0.9256064733497738
```

dari rfc.score tersebut diketahui bahwa akurasi model yang didapatkan adalah 95%. Kemudian sama seperti sebelumnya saya akan mevisualisasikan n_estimator pada model



Dari grafik tersebut diketahui bahwa dengan meningkatnya n_estimator, akurasi pengujian meningkat. Model mengevaluasi terbaik diperoleh pada n_estimators=100 yaitu akurasi 96,3%.

Kemudian saya akan melakukan pengecekan bagaimana model cocok untuk berbagai nilai 'max_features'



Dari grafik tersebut, terlihat bahwa model memberikan hasil terbaik untuk max_features=4 dengan akurasi mencapai lebih dari 97%.

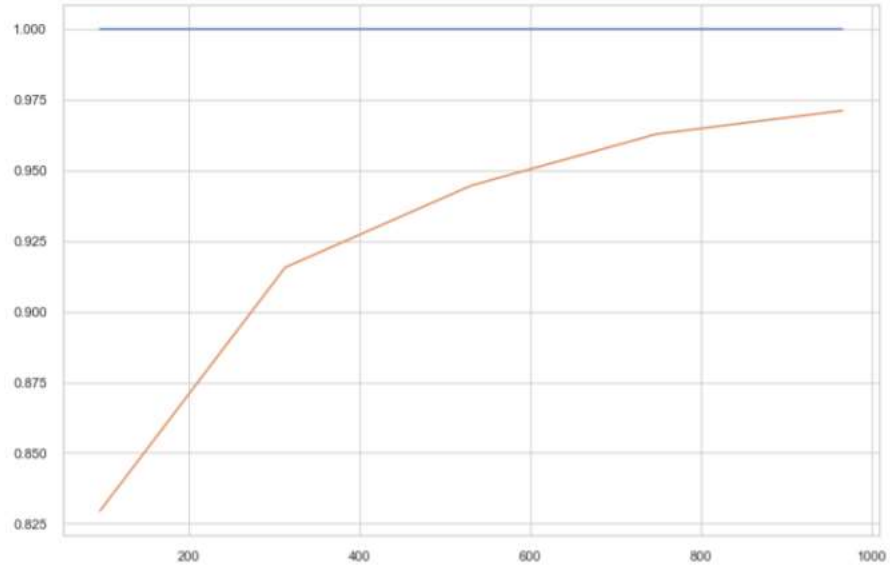
Cara lain yang lebih mudah adalah dengan menggunakan GridSearch untuk mendapatkan kombinasi parameter terbaik. Karena kumpulan data yang kecil, GridSearch akan membutuhkan lebih sedikit waktu.

In [82]:

```
print(grid.best_params_)
print(grid.best_score_)
```

```
{'criterion': 'gini', 'max_depth': 20, 'max_features': 4, 'max_leaf_nodes':
None}
0.978477961432507
```

dengan grid search score diperoleh RFC score mencapai akurasi 97,84%. kemudian sama seperti sebelumnya saya akan menjalankan dan mengevaluasi model tersebut dengan menggunakan learning_curve



Model akurasi data training adalah 1, tetapi akurasi data test kurang dari 1.

Untuk mengurangi varians, kita dapat melakukan penambahan data sample karena dari grafik sebelumnya diketahui bahwa semakin banyak sampel akan meningkatkan model. dan juga kita dapat megurangi atau menghilangkan variabel x. mari kita coba untuk mengurangi salah satu variabel x.

In [92]:

```
print(X.columns)
print(rfc.feature_importances_)
```

```
Index(['buying', 'maint', 'doors', 'persons', 'lug_boot', 'safety'], dtype
='object')
[0.18016727 0.16340398 0.06558715 0.24204797 0.08147323 0.26732039]
```

dari rfc.feature_importance tersebut diketahui bahwa variabel doors adalah variabel yang dianggap memiliki pengaruh paling kecil. jadi disini saya akan mencoba menjalankan model rfc dengan menghilangkan variabel doors.

In [85]:

```
X_train1, X_test1, y_train1, y_test1 = train_test_split(X[['buying', 'maint', 'persons', 'l
    y, test_size=0.3, random_state=42)

rfc1=RandomForestClassifier(n_estimators=100,criterion='entropy',max_features=4,max_depth=2
    max_leaf_nodes=None,n_jobs=-1)
rfc1.fit(X_train1,y_train1)
rfc1.score(X_test1,y_test1)
```

Out[85]:

```
0.928709055876686
```

dari hasil rfc.score tersebut diketahui bahwa dengan tanpa variabel doors akurasi yang diperoleh adalah 92% (atau tidak lebih baik dari model sebelumnya saat menggunakan variabel doors). sehingga dapat disimpulkan bahwa mengurangi variabel x tidak meningkatkan akurasi. satu-satunya cara untuk meningkatkan akurasi adalah menambah jumlah sampel.

KESIMPULAN

Random Forest Classifier adalah model yang paling sesuai untuk data ini dengan parameter berikut: n_estimators: 100 kriteria: entropy max_depth: 20 max_features: 4 max_leaf_nodes: Tidak ada. Hal tersebut dibuktikan dengan nilai akurasi yang diperoleh dari model tersebut sebesar 97,84%.