Laporan UTS - IBDA3111

Calvin Institute of Technology

Semester ganjil 2022/2023

Oleh

Stefannus Christian / 202000138 / IT & Big Data Analytics

Judul: Pembersihan Dataset Titanic dan membandingkan beberapa Model Klasifikasi

Disclaimer: Pada proposal data, saya menyebutkan bahwa saya akan membuat model Logistic Regression untuk memprediksi apakah suatu penumpang titanic selamat atau tidak. Pada laporan kali ini, saya akan membandingkan dan memaparkan 8 model lain selain Logistic Regression.

Latar Belakang

Tenggelamnya Titanic adalah salah satu *shipwreck* paling terkenal dalam sejarah. Pada tanggal 15 April 1912, selama pelayaran perdananya, RMS Titanic yang secara luas dianggap "tidak dapat tenggelam" tenggelam setelah bertabrakan dengan *iceberg*. Sayangnya, tidak ada cukup sekoci untuk semua penumpang, mengakibatkan kematian 1502 dari 2224 penumpang dan awak. Meskipun ada beberapa unsur keberuntungan yang terlibat dalam bertahan hidup, tampaknya beberapa kelompok orang lebih mungkin untuk bertahan hidup daripada yang lain.

Deskripsi Data

Terdapat 12 fitur dari dataset titanic yang saya ambil. Terdapat 11 fitur input dan 1 fitur

output.

1. PassengerID: Passenger ke i dari dataset (Unique Value).

2. Survived: Menandakan apakah suatu penumpang selamat atau tidak. 0 = tidak

selamat, 1 = selamat (Merupakan fitur output).

3. Pclass (Passenger Class): Divisi dari penumpang. Terdiri dari 3 kelas yaitu kelas 1, 2,

dan 3. Deskripsi Kelas: 1 → Upper Class, 2 → Middle Class, 3 → Lower Class.

4. Name: Nama dari penumpang.

5. Sex: Jenis kelamin penumpang.

6. Age: Umur dari penumpang.

7. SibSp (Sibling Spouse): Jumlah saudara kandung atau pasangan dari seorang penumpang di dalam kapal. Note: Mistresses and Fiances Ignored.

8. Parch: Jumlah orangtua atau anak dari seorang penumpang di dalam kapal.

9. Ticket: Nomor ticket dari penumpang.

10. Fare: Harga tiket kapal yang dibayar penumpang dalam British Pound pada saat itu.

11. Cabin: Nomor Cabin penumpang.

12. Embarked: Dari mana penumpang naik kapal. Ada tiga kemungkinan nilai untuk fitur ini yaitu Southampton, Cherbourg, dan Quennstown

Sumber Data

Data dapat diambil dari sumber berikutL
https://github.com/krishnaik06/EDA1/blob/master/titanic_train.csv

Dataset

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S

Dapat dilihat dari dataset bahwa terdapat data – data kosong dan banyak data kategorikal sehingga diperlukan prapemrosesan dan rekayasa data sebelum dimasukkan ke *Machine Learning Model.*

Prapemrosesan / Rekayasa Data

Mendrop Kolom PassengerID dan Ticket & Mengganti Nama Kolom Dataset

Mendrop Kolom PassengerID dan Ticket & Mengganti Nama Kolom Dataset

Kolom PassengerId tidak relevan terhadap dataset dan tidak ada informasi apa - apa yang dapat diperoleh dari fitur ticket

```
df.drop(['PassengerId'],axis=1,inplace=True)
df.drop(['Ticket'],axis=1,inplace=True)
df.columns = ['Survived','Class','Name','Sex','Age','#Spouse','#NumParents&Kids','Fare','Cabin','Embarked']
display(df.head())
```

✓ 0.7s

	Survived	Class	Name	Sex	Age	#Spouse	#NumParents&Kids	Fare	Cabin	Embarked
0	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	7.2500	NaN	S
1	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	71.2833	C85	C
2	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	7.9250	NaN	S
3	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	53.1000	C123	S
4	0	3	Allen, Mr. William Henry	male	35.0	0	0	8.0500	NaN	S

Kolom PassengerId isinya hanyalah id penumpang dengan kata lain menandakan penumpang ke – i sehingga tidak relevan bagi model dan dapat dibuang. Kemudian, saya juga mendrop kolom ticket karena isi dari kolom ticket adalah nomor ticket dari penumpang. Sulit untuk mendapatkan sesuatu yang berguna dari kolom tersebut, sehingga saya memutuskan untuk membuang kolom tersebut.

Menambahkan Fitur Family Size

Feature Engineering

Menambah Fitur Family Size

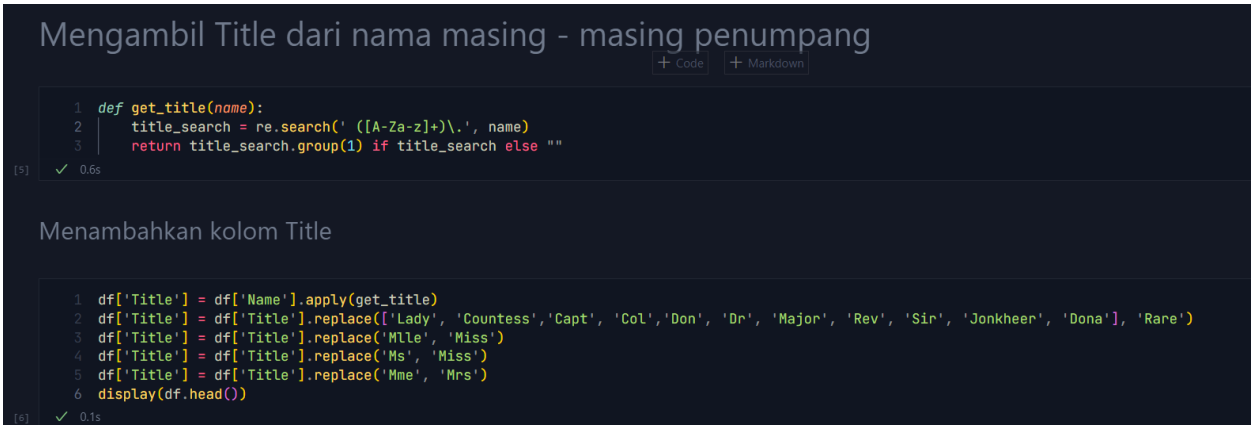
Family size adalah gabungan dari fitur jumlah spouse dan orangtua/anak yang terdapat pada kapal

```
df['FamilySize'] = df['#Spouse'] + df['#NumParents&Kids'] + 1
```

[14] ✓ 0.1s

Setelah melihat dari beberapa sumber, saya melihat ada satu sumber yang menambahkan fitur family size yaitu jumlah spouse ditambah dengan jumlah orangtua/anak yang terdapat pada kapal, sehingga saya memutuskan untuk menambahkan fitur ini ke dataset.

Menambahkan Title dari nama masing – masing penumpang



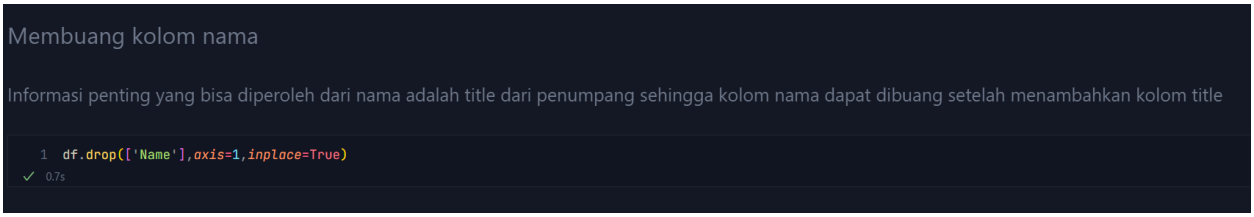
Name
Braund, Mr. Owen Harris
Cumings, Mrs. John Bradley (Florence Briggs Th...
Heikkinen, Miss. Laina
Futrelle, Mrs. Jacques Heath (Lily May Peel)
Allen, Mr. William Henry

Saya membuat fungsi menggunakan *regex (regular expression)* untuk mendeteksi *title* penumpang seperti Mr, Miss, Dr, dll. Tujuan saya melakukan hal ini adalah agar fitur nama tidak dibuang begitu saja. Saya melihat adanya titel – titel ini yang dapat diekstrak menjadi fitur baru yang mungkin berguna ketika pelatihan model nanti.

Dataset setelah menambahkan fitur Title

	Survived	Class	Name	Sex	Age	#Spouse	#NumParents&Kids	Fare	Cabin	Embarked	FamilySize	Title
0	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	7.2500	NaN	S	2	Mr
1	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	71.2833	C85	C	2	Mrs
2	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	7.9250	NaN	S	1	Miss
3	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	53.1000	C123	S	2	Mrs
4	0	3	Allen, Mr. William Henry	male	35.0	0	0	8.0500	NaN	S	1	Mr

Membuang Kolom Nama



Informasi penting yang dapat diambil dari nama hanyalah titel dari penumpang sehingga kolom nama dapat di buang.

Mengelompokkan Fitur *Fare* menjadi 4 kelompok

Saya mengelompokkan fitur *Fare* menjadi 4 kelompok karena fitur *fare* memiliki range data yang berbeda dari data – data lainnya. Data – data lainnya akan berbentuk kategorikal (0 – 1) dan fitur *fare* tidak berbentuk kategorikal, sehingga saya rasa bahwa fitur *fare* akan merusak pemodelan. Maka dari itu, memutuskan untuk mengelompokkan fitur *Fare* agar nantinya fitur ini dapat berbentuk kategorikal juga.

Dataset setelah merubah Fitur *Fare* menjadi 4 kelompok

Survived	Class	Sex	Age	#Spouse	#NumParents&Kids	Fare	Cabin	Embarked	FamilySize	Title
1	3	female	NaN	0	0	low_fare	NaN	Q	1	Miss
0	3	male	NaN	0	0	low_fare	NaN	C	1	Mr
1	3	female	NaN	0	0	low_fare	NaN	Q	1	Miss
0	3	male	NaN	0	0	low_fare	NaN	S	1	Mr
0	3	male	NaN	0	0	median_fare	NaN	S	1	Mr
...	...	...	...	...	...	...	...	...	...	...
0	3	male	NaN	0	0	low_fare	NaN	S	1	Mr
0	3	male	33.0	0	0	low_fare	NaN	S	1	Mr
0	3	female	22.0	0	0	median_fare	NaN	S	1	Miss
0	2	male	28.0	0	0	median_fare	NaN	S	1	Mr
0	3	male	25.0	0	0	low_fare	NaN	S	1	Mr

Metode Data Cleaning 1 – Membuang Data Duplikasi

Metode Data Cleaning 1 - Membuang Data Duplikasi

Menampilkan Data - Data Terduplikasi

Dapat dilihat dari kode dibawah bahwa terdapat **194** baris yang redundan / duplikat

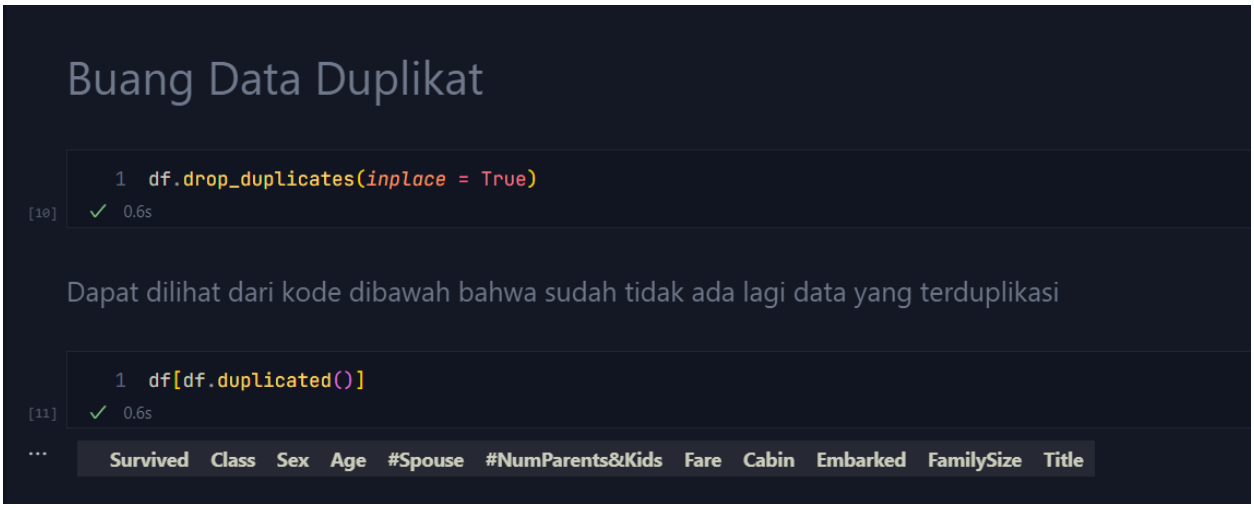
```
1 df[df.duplicated()]
```

✓ 0.7s

	Survived	Class	Sex	Age	#Spouse	#NumParents&Kids	Fare	Cabin	Embarked	FamilySize	Title
32	1	3	female	NaN	0	0	low_fare	NaN	Q	1	Miss
42	0	3	male	NaN	0	0	low_fare	NaN	C	1	Mr
47	1	3	female	NaN	0	0	low_fare	NaN	Q	1	Miss
76	0	3	male	NaN	0	0	low_fare	NaN	S	1	Mr
77	0	3	male	NaN	0	0	median_fare	NaN	S	1	Mr
...	...	...	...	...	...	...	...	...	...	...	...
878	0	3	male	NaN	0	0	low_fare	NaN	S	1	Mr
881	0	3	male	33.0	0	0	low_fare	NaN	S	1	Mr
882	0	3	female	22.0	0	0	median_fare	NaN	S	1	Miss
883	0	2	male	28.0	0	0	median_fare	NaN	S	1	Mr
884	0	3	male	25.0	0	0	low_fare	NaN	S	1	Mr

194 rows × 11 columns

Dapat dilihat dari kode dibawah bahwa terdapat **194** baris yang redundan / duplikat



Buang Data Duplikat

```
1 df.drop_duplicates(inplace = True)
```

[10] ✓ 0.6s

Dapat dilihat dari kode dibawah bahwa sudah tidak ada lagi data yang terduplikasi

```
1 df[df.duplicated()]
```

[11] ✓ 0.6s

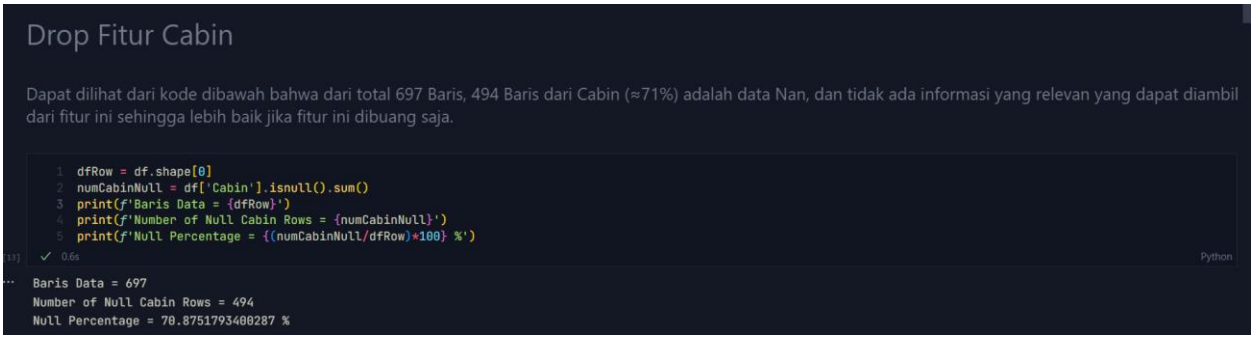
Survived	Class	Sex	Age	#Spouse	#NumParents&Kids	Fare	Cabin	Embarked	FamilySize	Title

Terdapat 194 baris data duplikasi yang saya buang. Saya membuang duplikasi data sehingga model dapat digeneralisasi dengan lebih baik ke *dataset*.

Metode Data Cleaning 2 – Imputasi Data

Imputasi data dilakukan karena ada 4 fitur dataset yang memiliki data – data null.

Drop Fitur Cabin



Dari 697 baris data tersisa, 494 (≈71%) baris dari fitur *Cabin* merupakan data kosong, dan sulit untuk melakukan imputasi pada fitur ini sehingga saya memutuskan untuk membuang fitur *Cabin*.

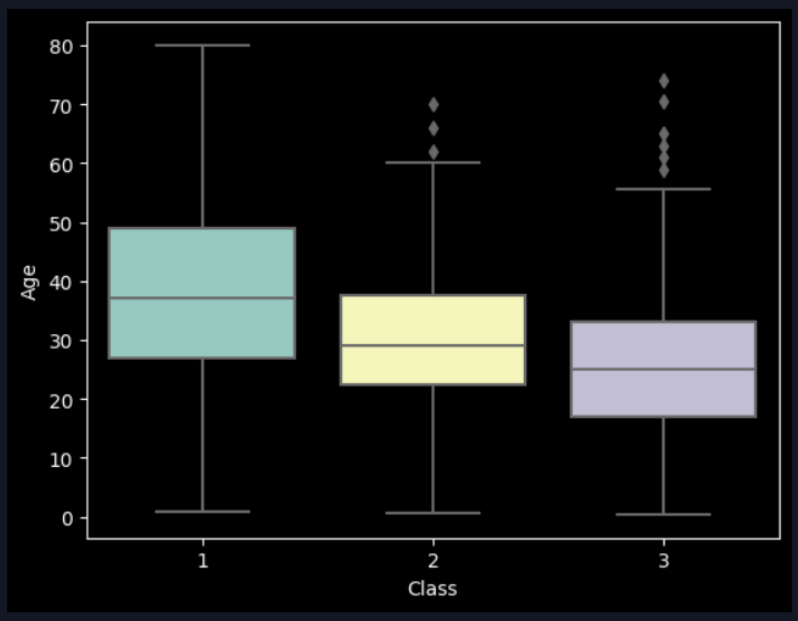
Imputasi Fitur Age berdasarkan kelas Penumpang

Menampilkan BoxPlot rata - rata umur penumpang berdasarkan Class

1 adalah kelas bawah, 2 adalah kelas menengah, dan 3 adalah kelas atas

```
1 sns.boxplot(x='Class',y='Age',data=df)
2 plt.show()
```

[15] ✓ 0.1s



Untuk imputasi fitur *Age*, saya memutuskan untuk menggunakan imputasi rata – rata. Tetapi, rata – rata yang saya gunakan bukan satu rata – rata saja tetapi tiga rata – rata. Tiga rata – rata ini saya klasifikasikan berdasarkan class penumpang (penjelasan di gambar bawah).

Membuat function untuk mengimputasi fitur age dengan rata-rata dari class penumpang tersebut

Misalkan rata-rata dari class 1, 2, 3 adalah 35, 25,dan 20. Algoritma akan looping dataset dan mencari cell yang memiliki nilai null, jika ketemu cell yang nilai nya null, algoritma akan check class dari penumpang tersebut. Jika class dari penumpang tersebut adalah class 2, maka cell kosong tersebut akan diisi dengan rata-rata umur dari penumpang class 2 yaitu 25 (misalkan)

Berikut adalah *source code* yang saya buat untuk mengimputasi data – data null pada fitur *age* berdasarkan *Passenger Class*.

```python
# Membuat Subset dataframe yang isinya kelas penumpang dan Age untuk memudahkan
class_age_df = df[['Class','Age']]

# Fungsi untuk mendapatkan rata - rata umur penumpang per kelas
def get_age_mean_by_class(df):
    low_class_mean = round(df[(df['Class']==1)].mean())[1]
    med_class_mean = round(df[(df['Class']==2)].mean())[1]
    high_class_mean = round(df[(df['Class']==3)].mean())[1]
    return [low_class_mean, med_class_mean, high_class_mean]

# Algoritma yang dijelaskan pada markdown diatas
def age_mean_imputation_base_on_class(age_class_df):
    # Membuat list yang isinya rata-rata umur penumpang per kelas
    age_mean_by_class = get_age_mean_by_class(class_age_df)
    Age = age_class_df[0]
    Class = age_class_df[1]
    # Jika ketemu cell yang kosong maka cell akan diimputasi berdasarkan rata-rata umur pada kelas penumpang tersebut
    if pd.isnull(Age):
        if Class == 1: return age_mean_by_class[0]
        elif Class == 2: return age_mean_by_class[1]
        else: return age_mean_by_class[2]

    # Jika tidak kosong maka return age sesungguhnya
    else: return Age

# Apply fungsi diatas pada dataframe
df['Age'] = df[['Age','Class']].apply(age_mean_imputation_base_on_class,axis=1)
```

✓ 0.8s

Mengecek apakah kolom age sudah terimputasi dengan baik

Dapat dilihat dari kode dibawah bahwa sudah tidak ada lagi data null pada fitur Age

```python
df['Age'].isnull().sum()
```

[17] ✓ 0.4s

```
0
```

Setelah fitur *Age* diimputasi, dapat dilihat bahwa sudah tidak ada lagi baris yang null pada fitur *Age*.

Mengelompokkan Fitur Age menjadi 4 Kelompok

Mengelompokkan Fitur Age menjadi 4 kelompok

Sama seperti kolom fare, setelah fitur age diimputasi, fitur age dikelompokkan menjadi 4 kelompok yaitu Children, Teenange, Adult, dan Elder

```python
df['Age'] = pd.cut(df['Age'], bins=[0,12,20,40,120], labels=['Children','Teenage','Adult','Elder'])
display(df)
```

[18] ✓ 0.7s

Saya mengelompokkan Fitur *Age* karena sama dengan *Fare* takutnya karena fitur *Age* tidak bersifat kategorikal (0-1, dll) maka saya takutnya fitur ini akan "merusak" pelatihan model sehingga saya mengelompokkan fitur ini sehingga nantinya dapat dilakukan *One Hot Encoding*.

	Survived	Class	Sex	Age	#Spouse	#NumParents&Kids	Fare	Embarked	FamilySize	Title
0	0	3	male	Adult	1	0	low_fare	S	2	Mr
1	1	1	female	Adult	1	0	high_fare	C	2	Mrs
2	1	3	female	Adult	0	0	median_fare	S	1	Miss
3	1	1	female	Adult	1	0	high_fare	S	2	Mrs
4	0	3	male	Adult	0	0	median_fare	S	1	Mr
...	...	...	...	...	...	...	...	...	...	...
886	0	2	male	Adult	0	0	median_fare	S	1	Rare
887	1	1	female	Teenage	0	0	average_fare	S	1	Miss
888	0	3	female	Adult	1	2	average_fare	S	4	Miss
889	1	1	male	Adult	0	0	average_fare	C	1	Mr
890	0	3	male	Adult	0	0	low_fare	Q	1	Mr

697 rows × 10 columns

Imputasi Kolom Embarked

Hanya terdapat 2 baris dari fitur *Embarked* yang memiliki data null sehingga saya akan mengimputasi kolom embarked ini menggunakan modus atau data terbanyak. Data

terbanyak fitur *Embarked* adalah *Southampton* sehingga saya akan mengisi 2 baris yang kosong ini dengan *Southampton.*

Mengatasi Fitur Fare yang memiliki null *values*

Saya akan mengatasi null *values* pada fitur *fare* dengan One Hot Encoding. One Hot Encoding akan mengimputasi sendiri data – data yang null pada fitur *fare.*

One Hot Encoding

Saya melakukan *One Hot Encoding* untuk fitur – fitur *Sex, Title, Age, Embarked, dan Fare*.

Berikut adalah tampilan dataset setelah dilakukan One Hot Encoding.

	Survived	Class	#Spouse	#NumParents&Kids	FamilySize	Sex_male	Title_Miss	Title_Mr	Title_Mrs	Title_Rare	Age_Teenage	Age_Adult	Age_Elder	Embarked_Q	Embarked_S	Fare_median_fare
0	0	3	1	0	2	1	0	1	0	0	0	1	0	0	1	0
1	1	1	1	0	2	0	0	0	1	0	0	1	0	0	0	0
2	1	3	0	0	1	0	1	0	0	0	0	1	0	0	1	1
3	1	1	1	0	2	0	0	0	1	0	0	1	0	0	1	0
4	0	3	0	0	1	1	0	1	0	0	0	1	0	0	1	1

Summary DataFrame setelah di One Hot Encode

Summary dari DataFrame

```
1 df.describe()
```

✓ 0.7s Python

	Survived	Class	#Spouse	#NumParents&Kids	FamilySize	Sex_male	Title_Miss	Title_Mr	Title_Mrs	Title_Rare	Age_Teenage	Age_Adult	Age_Elder	Embarked_Q	Embarke
count	697.000000	697.000000	697.000000	697.000000	697.000000	697.000000	697.000000	697.000000	697.000000	697.000000	697.000000	697.000000	697.000000	697.000000	697.000
mean	0.449067	2.166428	0.595409	0.470588	2.065997	0.592539	0.226686	0.507891	0.176471	0.032999	0.126255	0.566714	0.209469	0.061693	0.721
std	0.497756	0.864942	1.065327	0.875634	1.607706	0.491715	0.418988	0.500297	0.381494	0.178761	0.332375	0.495885	0.407222	0.240770	0.448
min	0.000000	1.000000	0.000000	0.000000	1.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000
25%	0.000000	1.000000	0.000000	0.000000	1.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000
50%	0.000000	2.000000	0.000000	0.000000	1.000000	1.000000	0.000000	1.000000	0.000000	0.000000	0.000000	1.000000	0.000000	0.000000	1.000
75%	1.000000	3.000000	1.000000	1.000000	3.000000	1.000000	0.000000	1.000000	0.000000	0.000000	0.000000	1.000000	0.000000	0.000000	1.000
max	1.000000	3.000000	8.000000	6.000000	11.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000

Mengecek tipe data dari semua fitur

Machine Learning model tidak dapat mengerti data - data kategorikal berbentuk string atau
bentuk integer/float. Dapat dilihat dari kode dibawah bahwa seluruh dataset sudah dalam b

```
[26] 1 print(df.dtypes)
```

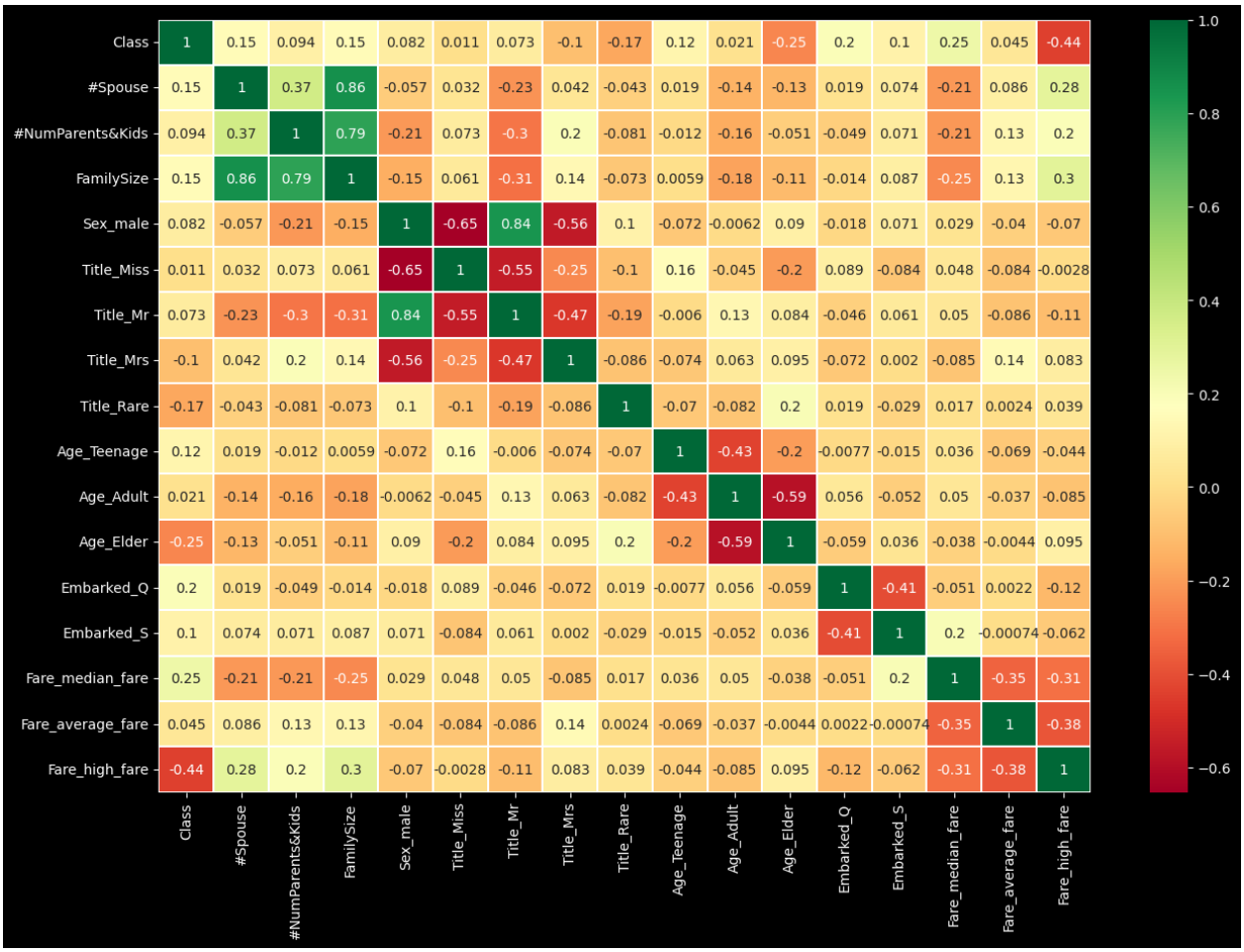
✓ 0.9s

```
Survived              int64
Class                 int64
#Spouse               int64
#NumParents&Kids      int64
FamilySize            int64
Sex_male              uint8
Title_Miss            uint8
Title_Mr              uint8
Title_Mrs             uint8
Title_Rare            uint8
Age_Teenage           uint8
Age_Adult             uint8
Age_Elder             uint8
Embarked_Q            uint8
Embarked_S            uint8
Fare_median_fare      uint8
Fare_average_fare     uint8
Fare_high_fare        uint8
dtype: object
```

Gambar diatas adalah untuk pengecekan tipe data dari dataframe setelah dilakukan one hot encoding. Diperlukan semua data – data numerik agar model dapat dilatih sehingga perlu dipastikan terlebih dahulu kalau semua fitur dataframe adalah data – data numerik.

Correlation Matrix

Berikut adalah matriks korelasi dataframe setelah dilakukan one hot encoding.



RFE Feature Selection

Menampilkan Estimator terbaik beserta jumlah fitur nya

```
1  bestEstimatorLogReg = getBestRFEEstimatorWithNumFeatures(logregResults)
2  printBestRFEEstimatorWithNumFeaturesName(bestEstimatorLogReg,modelName)
```

[33] ✓ 0.4s

```
RFE Estimator terbaik untuk model Logistic Regression adalah Gradient Boosting dengan 11 jumlah fitur.
Akurasi = 78.05003450655623%
```

Membandingkan Performa 9 Model Klasifikasi sebelum dan sesudah dilakukan RFE Feature Selection

Membuat Dataframe newDfRFE yang isinya adalah fitur-fitur yang dipilih oleh Estimator Logistic Regression dengan 13 jumlah fitur input dan 1 fitur output

```
selectedFeatures = getSelectedFeatures(logisticRegressionModel,bestEstimatorLogReg,rfeSupport)
newDfRFE = df.iloc[:,selectedFeatures]
newDfRFE
```

[34] ✓ 0.9s Python

	Survived	Class	#Spouse	FamilySize	Sex_male	Title_Mr	Title_Rare	Age_Adult	Age_Elder	Embarked_S	Fare_median_fare	Fare_average_fare
0	0	3	1	2	1	1	0	1	0	1	0	0
1	1	1	1	2	0	0	0	1	0	0	0	0
2	1	3	0	1	0	0	0	1	0	1	1	0
3	1	1	1	2	0	0	0	1	0	1	0	0
4	0	3	0	1	1	1	0	1	0	1	1	0
...	...	...	...	...	...	...	...	...	...	...	...	...
886	0	2	0	1	1	0	1	1	0	1	1	0
887	1	1	0	1	0	0	0	0	0	1	0	1
888	0	3	1	4	0	0	0	1	0	1	0	1
889	1	1	0	1	1	1	0	1	0	0	0	1
890	0	3	0	1	1	1	0	1	0	0	0	0

697 rows × 12 columns

Detail kode dapat dilihat pada file ipynb. RFE estimator terbaik yang saya dapatkan adalah Gradient Boosting estimator dengan model Logistic Regression.

Model yang memiliki improvement akurasi setelah RFE feature selection

No	Model	Akurasi (FS❌)	Akurasi (FS✅)
1	Logistic Regression	75.714 %	79.524 %
2	KNN	80.476 %	81.429 %
3	SVM	78.571 %	80.952 %
4	Naive-Bayes	77.619 %	79.524 %
5	Decision Tree	79.524 %	80.952 %
6	RandomForest	80.0 %	80.476 %
7	Linear Discriminant Analysis	78.571 %	80.0 %

Model yang memiliki penurunan akurasi setelah RFE feature selection

No	Model	Akurasi (FS❌)	Akurasi (FS✅)
1	Ada Boost	78.571 %	80.0 %
2	Gradient Boosting	81.905 %	80.476 %

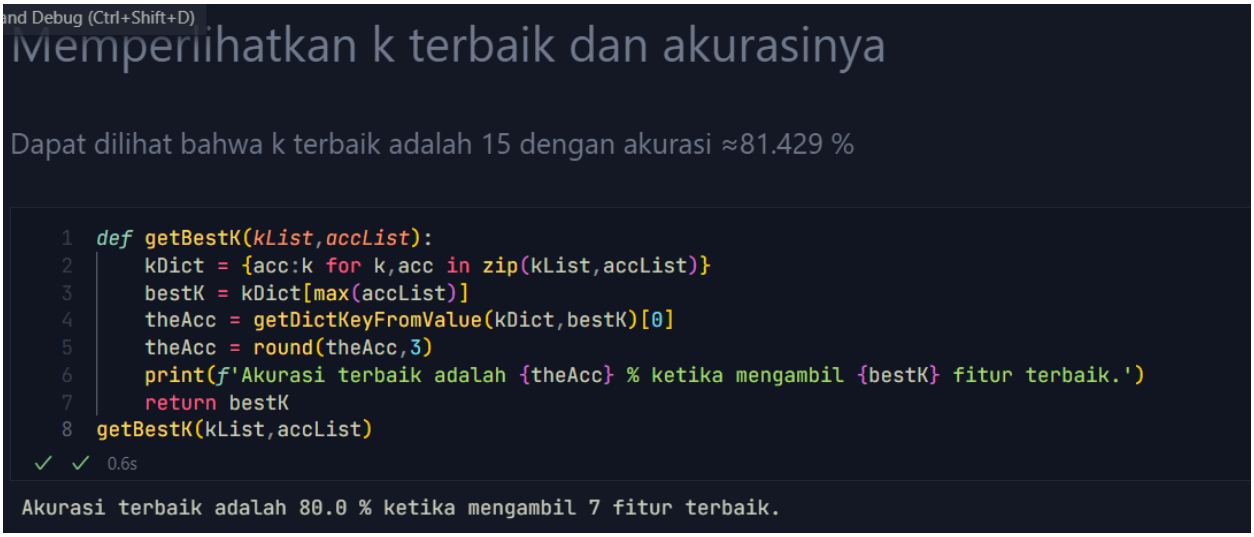
Ini harusnya ada pada tabel peningkatan akurasi.

Dapat dilihat dari kode diatas bahwa dari ke sembilan model, terdapat 8 model yang memiliki peningkatan akurasi ketika dilakukan RFE Feature estimator dengan menggunakan estimator Gradient Boosting dan mengambil 11 fitur.

Dapat dilihat bahwa model KNN dengan 4 neighbors memiliki akurasi terbaik setelah dilakukan RFE Feature Selection yaitu sebesar 81.429%.

Mutual Information Feature Selection

Detail kode dapat dilihat pada file ipynb



Saya menggunakan model KNN dengan n neighbors = 4 berdasarkan hasil dari RFE feature selection dan dapat dilihat bahwa metode mutual information mengambil 7 fitur terbaik dengan akurasi 80%.

Solusi

- Gunakan model KNN untuk dataset Titanic.
- Gunakan Feature Selection RFE dengan estimator Gradient Boosting dan Model Logistic Regression dan lakukan for loop untuk membandingkan semua n_features_to_select dan gunakan lah hasil akurasi yang terbaik
- Lakukan lah one hot encoding dan pecahlah kolom age dan fare seperti yang saya lakukan diatas
- Jangan langsung Drop Fitur Nama tetapi ambilah title dari nama tersebut
- Buanglah Kolom Cabin, PassengerId, dan Ticket

Kesimpulan

- Model KNN dengan n_neighbors = 4 merupakan model terbaik untuk mengklasifikasi apakah suatu penumpang selamat atau tidak dari dataset titanic.
- Akurasi Model KNN setelah dilakukan RFE Feature Selection dengan memilih 11 fitur dan estimator gradient boosting adalah 81%.
- Feature selection meningkatkan akurasi model untuk titanic Dataset
- RFE Feature Selection lebih baik daripada mutual Information Feature Selection

Komitmen Integritas

“Di hadapan TUHAN yang hidup, saya menegaskan bahwa saya tidak memberikan maupun menerima bantuan apapun—baik lisan, tulisan, maupun elektronik—di dalam ujian ini selain daripada apa yang telah diizinkan oleh pengajar, dan tidak akan menyebarkan baik soal maupun jawaban ujian kepada pihak lain.”

Stefannus Christian 202000138