Predict Student Grant Recommendations

Objective

You have history student performance name student_records.csv , from that's let we start study Machine Learning.

Data Retrivial

In [1]:
```
import pandas as pd
# turn of warning maessages
pd.options.mode.chained_assignment=None #default='warn'

#get data
df = pd.read_csv('student_records.csv')
df
```

Out[1]:

	Name	OverallGrade	Obedient	ResearchScore	ProjectScore	Recommend
0	Henry	A	Y	90	85	Yes
1	John	C	N	85	51	Yes
2	David	F	N	10	17	No
3	Holmes	B	Y	75	71	No
4	Marvin	E	N	20	30	No
5	Simon	A	Y	92	79	Yes
6	Robert	B	Y	60	59	No
7	Trent	C	Y	75	33	No

Data preparation

Based data above we stil have data error ,missing value so we start scalling in this section Fiture extraction and engineering

In [2]:
```
# get fitures and corresponding outcomes
feature_names = ['OverallGrade','Obedient','ResearchScore','ProjectScore']
training_features = df[feature_names]
outcome_name = ['Recommend']
outcome_labels = df[outcome_name]
```

In [3]:
```
# view features
training_features
```

Out[3]:

	OverallGrade	Obedient	ResearchScore	ProjectScore
0	A	Y	90	85
1	C	N	85	51
2	F	N	10	17
3	B	Y	75	71
4	E	N	20	30
5	A	Y	92	79
6	B	Y	60	59
7	C	Y	75	33

Melihat data set recommendation outcome labels for each student

In [4]:
```
# view outcome labels
outcome_labels
```

Out[4]:

	Recommend
0	Yes
1	Yes
2	No
3	No
4	No
5	Yes
6	No
7	No

Let’s separate out our available feature based on their type (numerical and categorical)

In [5]:
```
# list down feature based on type
numeric_feature_names = ['ResearchScore','ProjectScore']
categorical_feature_names = ['OverallGrade','Obedient']
```

Use standard scalar from sckit-learn to scale or normalize out two numeric score based attribute

In [6]:
```
from sklearn.preprocessing import StandardScaler
ss = StandardScaler()

# fit scalar on numeric feature
ss.fit(training_features[numeric_feature_names])

# fit  numeric feature now
training_features[numeric_feature_names] = ss.transform(training_features[numeric_feature_names])

# view update features
training_features
```

Out[6]:

	OverallGrade	Obedient	ResearchScore	ProjectScore
0	A	Y	0.899583	1.376650
1	C	N	0.730648	-0.091777
2	F	N	-1.803390	-1.560203
3	B	Y	0.392776	0.772004
4	E	N	-1.465519	-0.998746
5	A	Y	0.967158	1.117516
6	B	Y	-0.114032	0.253735
7	C	Y	0.392776	-0.869179

Melihat feature data set categorical variables

In [9]:
```
training_features = pd.get_dummies(training_features,columns=categorical_feature_names)

# view newly engineering features
training_features
```

Out[9]:

	ResearchScore	ProjectScore	OverallGrade_A	OverallGrade_B	OverallGrade_C	OverallGrade_E	OverallGrade_F	Obedient_N	Obedient_Y
0	0.899583	1.376650	1	0	0	0	0	0	1
1	0.730648	-0.091777	0	0	1	0	0	1	0
2	-1.803390	-1.560203	0	0	0	0	1	1	0
3	0.392776	0.772004	0	1	0	0	0	0	1
4	-1.465519	-0.998746	0	0	0	1	0	1	0
5	0.967158	1.117516	1	0	0	0	0	0	1
6	-0.114032	0.253735	0	1	0	0	0	0	1
7	0.392776	-0.869179	0	0	1	0	0	0	1

In [10]:
```
# get list of new categorical features
categorical_engineered_features = list(set(training_features.columns)-set(numeric_feature_names))
```

Kita lanjutkan pemodelan data

Disini kita mulai membangun clasifikasi sederhana supervised model berdasarkan fitures kita dengan menggunakan logic algoritma regression .

In [13]:
```
# Build Modelling
from sklearn.linear_model import LogisticRegression
import numpy as np

# fit the model
lr = LogisticRegression()
model = lr.fit(training_features,np.array(outcome_labels['Recommend']))

# view model parameters
model
```

Out[13]:
```
LogisticRegression()
```

Evaluasi model (jika bingung membaca hasil ini , baca Kembali theory statistic lagi ! atau perdalam di Chapter 5 nanti)

In [14]:
```
# Simple evaluation on training data
pred_labels = model.predict(training_features)
actual_labels = np.array(outcome_labels['Recommend'])

# evaluate model performance
from sklearn.metrics import accuracy_score
from sklearn.metrics import classification_report

print('Accuracy:' ,float(accuracy_score(actual_labels,pred_labels))*100,'%')
print('Classification Stats:')
print(classification_report(actual_labels,pred_labels))
```

```
Accuracy: 100.0 %
Classification Stats:
              precision    recall  f1-score   support

          No       1.00      1.00      1.00         5
         Yes       1.00      1.00      1.00         3

    accuracy                           1.00         8
   macro avg       1.00      1.00      1.00         8
weighted avg       1.00      1.00      1.00         8
```

Model deployment

SEjauh tahapan ini kita sudah memiliki model dari pengolahan data set tersebut dan selanjutnya kita gunaan acuan model tersebut dalam mesin server kita. Kita butuh menyimpan scalar object tsb guna di isi datasample yang baru.

In [15]:
```
import joblib
import os

# save model to be deployed on your server machine learning
if not os.path.exists('Model'):
    os.mkdir('Model')
if not os.path.exists('Scaler'):
    os.mkdir('Scaler')

joblib.dump(model,r'Model/model.pickle')
joblib.dump(ss,r'Scaler/scaler.pickle')
```

Out[15]:
```
['Scaler/scaler.pickle']
```

Prediction in action

Kita sampai sejauh inisudah siap untuk mulai melaksanakan proses prediction melalui proses pembuatan dan membangun model yang baru. Untuk memulai prediksi kita perlu meload model kita dalam memory server , berikut coding phyton tersebut

In [16]:
```
# load model and scaler object on memory server
model = joblib.load(r'Model/model.pickle')
scaler = joblib.load(r'Scaler/scaler.pickle')
```

Kita akan uji coba kan model yang sudah kita buat ini untuk melakukan prediksi dari 2 record data mahasiswa berikut ini apakah mereka mendapat rekomendasi beasiswa tersebut.

In [17]:
```
## data retrieval
new_data = pd.DataFrame([{'Name': 'Nathan','OverallGrade':'F','Obedient':'N','ResearchScore':30,'ProjectScore':20},
                         {'Name':'Burhanudin','OverallGrade':'A','Obedient':'Y','ResearchScore':78,'ProjectScore':80}])
new_data = new_data[['Name','OverallGrade','Obedient','ResearchScore','ProjectScore']]
new_data
```

Out[17]:

	Name	OverallGrade	Obedient	ResearchScore	ProjectScore
0	Nathan	F	N	30	20
1	Burhanudin	A	Y	78	80

Update data set for new student

In [19]:
```
## data preparation
prediction_features = new_data[feature_names]

# scaling
prediction_features[numeric_feature_names] = scaler.transform(prediction_features[numeric_feature_names])

# engineering categorical variables
prediction_features = pd.get_dummies(prediction_features,columns=categorical_feature_names)

# view feature set
prediction_features
```

Out[19]:

	ResearchScore	ProjectScore	OverallGrade_A	OverallGrade_F	Obedient_N	Obedient_Y
0	-1.127647	-1.430636	0	1	1	0
1	0.494137	1.160705	1	0	0	1

Sekarang kita telah memiliki new features untuk mahasiswa yang baru , dapat di saksikan new feature tersebut menghilangkan category Grade seperti B,C, dan E dan menghasilkan Feature final dari new mahasiswa tersebut

In [20]:
```
# add missing categorical feature columns
current_categorical_engineered_features = set(prediction_features.columns) - set(numeric_feature_names)

missing_features = set(categorical_engineered_features) - current_categorical_engineered_features

for feature in missing_features:
    # add zeros since absent in this data samples
    prediction_features[feature]=[0] * len(prediction_features)

# view final feature set
prediction_features
```

Out[20]:

	ResearchScore	ProjectScore	OverallGrade_A	OverallGrade_F	Obedient_N	Obedient_Y	OverallGrade_E	OverallGrade_B	OverallGrade_C
0	-1.127647	-1.430636	0	1	1	0	0	0	0
1	0.494137	1.160705	1	0	0	1	0	0	0

Dari sekarang feature lengkap kita akan siap untuk kedua mahasiswa tersebut. Mari kita lanjutkan meletakan model kita pada tahapan test guna mendapatkan hasil prediksi rekomendasi yang bisa di berikan terkait beasiswa mahasiswa tersebut

In [21]:
```
# predic using model
predictions = model.predict(prediction_features)

## display result
new_data['Recomend']= predictions
new_data
```

Out[21]:

	Name	OverallGrade	Obedient	ResearchScore	ProjectScore	Recomend
0	Nathan	F	N	30	20	No
1	Burhanudin	A	Y	78	80	Yes

Contoh tahapan case sederhana ini memberikan ilustrasi prediksi rekomnasi dalam machine learning.