Real-World Case Study: Predicting Student Grant Recommendations

- **Aluno:** Matheus Freitas Martins
- **Matrícula:** ES111281

Recuperação de Dados

In [1]:
```
import pandas as pd
# turn of warning messages
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

Preparação dos dados

In [2]:
```
#get features 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

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

In [5]:
```
# list down features based on type
numeric_feature_names = ['ResearchScore', 'ProjectScore']
```

```
categorical_feature_names = ['OverallGrade', 'Obedient']
```

Modelando

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

# fit scaler on numeric features
ss.fit(training_features[numeric_feature_names])

# scale numeric features now
training_features[numeric_feature_names] = ss.transform(training_features[numeric_feature_names])

# view updated featureset
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

In [7]:
```
training_features = pd.get_dummies(training_features, columns=categorical_feature_names)

# view newly engineering features
training_features
```

Out[7]:

	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 [8]:
```
# get list of new categorical features
categorical_engineered_features = list(set(training_features.columns) - set(numeric_feature_names))
```

In [9]:
```
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[9]:
```
▾ LogisticRegression
LogisticRegression()
```

In [10]:
```
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='liblinear', tol=0.0001, verbose=0,
                   warm_start=False)
```

Out[10]:
```
▾ LogisticRegression
LogisticRegression(multi_class='ovr', n_jobs=1, solver='liblinear')
```

Avaliação do modelo

In [11]:
```
# 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
```

Implantação do modelo

In [12]:
```
#from sklearn.externals import joblib
# a partir da versão 0.23 do scikit-learn, o módulo joblib foi removido de sklearn.externals.
# É necessário instalar e importar o joblib diretamente.
!pip install joblib
import joblib
import os
# save models to be deployed on your server
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')
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Requirement already satisfied: joblib in /usr/local/lib/python3.9/dist-packages (1.1.1)
```

Out[12]:
```
['Scaler/scaler.pickle']
```

Predição em ação

In [13]:
```
# load model and scaler objects
model = joblib.load(r'Model/model.pickle')
scaler = joblib.load(r'Scaler/scaler.pickle')
```

In [14]:
```
## data retrieval
new_data = pd.DataFrame([{'Name': 'Nathan', 'OverallGrade': 'F',
 'Obedient': 'N', 'ResearchScore': 30, 'ProjectScore': 20},
 {'Name': 'Thomas', 'OverallGrade': 'A',
  'Obedient': 'Y', 'ResearchScore': 78, 'ProjectScore': 80}])

new_data = new_data[['Name', 'OverallGrade', 'Obedient', 'ResearchScore', 'ProjectScore']]
new_data
```

Out[14]:

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

In [15]:
```
## 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[15]:

	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

In [16]:
```
# 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 feature is absent in these data samples
  prediction_features[feature] = [0] * len(prediction_features)

# Esta linha garante que a ordem das colunas em prediction_features seja exatamente a mesma que a ordem das colunas em training_f
prediction_features = prediction_features[training_features.columns]

# view final feature set
prediction_features
```

Out[16]:

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

In [17]:

```
## predict using model
predictions = model.predict(prediction_features)

## display results
new_data['Recommend'] = predictions
new_data
```

Out[17]:

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