Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования

«Московский государственный технический университет

имени Н.Э. Баумана (национальный исследовательский университет)»

(МГТУ им. Н.Э. Баумана)

Факультет «Информатика и системы управления»

Кафедра «Системы обработки информации и управления»

Курс «Технологии машинного обучения»

Отчёт по лабораторной работе №4

«Линейные модели, SVM и деревья решений**»**

Выполнил:

студент группы ИУ5-63Б

Иванченко Максим

Проверил:

к.т.н., доц., Ю. Е. Гапанюк

2024 г.

Задание:

1. Выберите набор данных (датасет) для решения задачи классификации или регрессии.
2. В случае необходимости проведите удаление или заполнение пропусков и кодирование категориальных признаков.
3. С использованием метода train_test_split разделите выборку на обучающую и тестовую.
4. Обучите следующие модели:
   - одну из линейных моделей (линейную или полиномиальную регрессию при решении задачи регрессии, логистическую регрессию при решении задачи классификации);
   - SVM;
   - дерево решений.
5. Оцените качество моделей с помощью двух подходящих для задачи метрик. Сравните качество полученных моделей.
6. Постройте график, показывающий важность признаков в дереве решений.
7. Визуализируйте дерево решений или выведите правила дерева решений в текстовом виде.

Используемый датасет:
https://www.kaggle.com/code/mayankkestwal10/graduate-admissions-predictive-modelling/input.

```
import numpy as np
import pandas as pd
import graphviz
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
from sklearn.tree import DecisionTreeClassifier, export_graphviz
from sklearn.model_selection import GridSearchCV
from IPython.core.display import HTML
from sklearn.tree import export_text
from sklearn.metrics import accuracy_score
from sklearn.metrics import confusion_matrix
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import roc_curve, roc_auc_score
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
sns.set(style="ticks")
```

```
[2] dataset = pd.read_csv('./Admission_Predict.csv')
    dataset.head()
```

	Serial-No.	GRE-Score	TOEFL-Score	University Rating	SOP	LOR	CGPA	Research	Chance-of-Admit
0	1	337	118	4	4.5	4.5	9.65	1	0.92
1	2	324	107	4	4.0	4.5	8.87	1	0.76
2	3	316	104	3	3.0	3.5	8.00	1	0.72
3	4	322	110	3	3.5	2.5	8.67	1	0.80
4	5	314	103	2	2.0	3.0	8.21	0	0.65

Next steps: View recommended plots

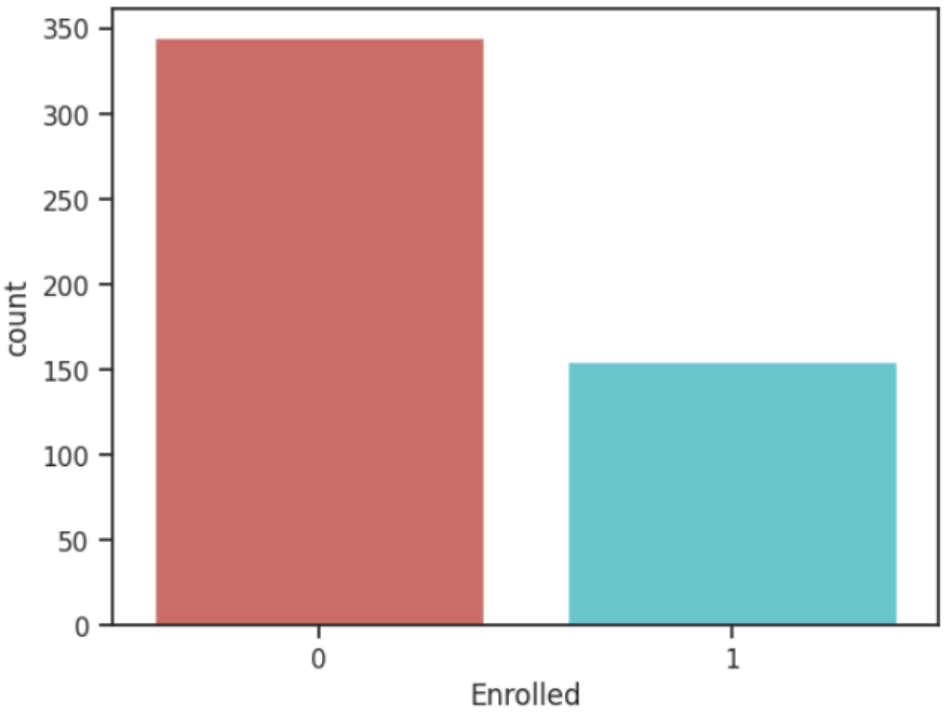
```
dataset.isna().sum()
```

```
Serial-No.          0
GRE-Score           0
TOEFL-Score         0
University Rating   0
SOP                 0
LOR                 0
CGPA                0
Research            0
Chance-of-Admit     0
dtype: int64
```

```
[4] dataset['Enrolled'] = np.where(dataset['Chance-of-Admit'] >= 0.80, 1, 0)
    dataset.drop(['Chance-of-Admit'], axis=1, inplace=True)
```

```
[5] sns.countplot(x='Enrolled', data=dataset, palette='hls')
    plt.show()
```

```
sns.countplot(x='Enrolled', data=dataset, palette='hls')
```



```
# Разделение на объекты-признаки и целевой признак
X = dataset.iloc[:, :-1].values
y = dataset.iloc[:, -1].values
```

[7]
```
# Формирование обучающей и тестовой выборки
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state = 1)
```

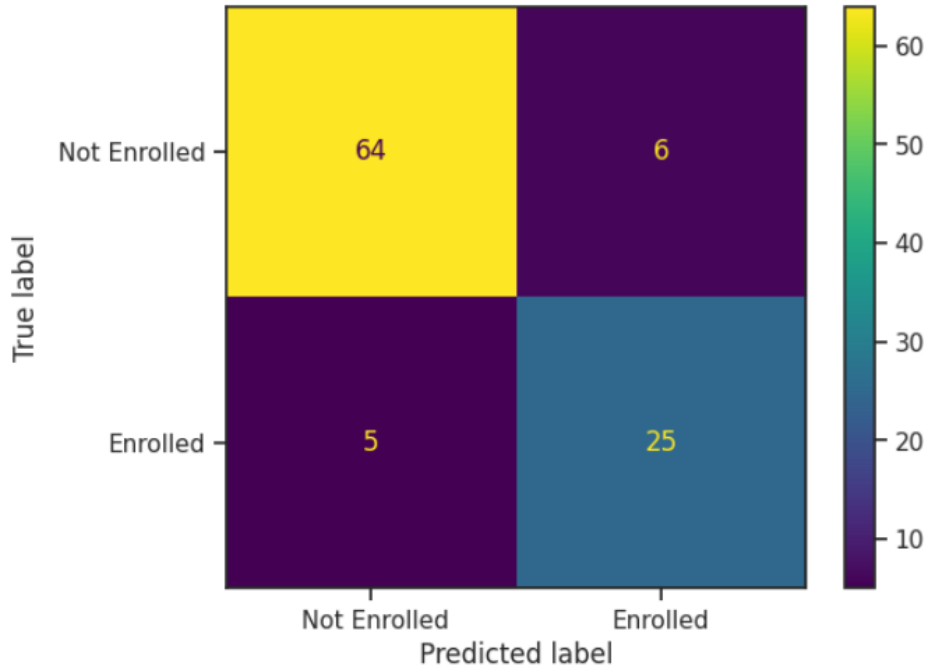
[8]
```
logreg = LogisticRegression()
logreg.fit(X_train, y_train)
y_pred_test_logreg = logreg.predict(X_test)
y_pred_train_logreg = logreg.predict(X_train)
ac1 = accuracy_score(y_train, y_pred_train_logreg), accuracy_score(y_test, y_pred_test_logreg)
ac1
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_logistic.py:458: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
(0.9, 0.89)
```

[9]
```
cm1 = confusion_matrix(y_test, y_pred_test_logreg, labels = logreg.classes_)
disp = ConfusionMatrixDisplay(confusion_matrix=cm1, display_labels=[' Not Enrolled', 'Enrolled'])
disp.plot()
```

```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x7fdd7a1e1030>
```



[10]
```
svc = SVC(kernel='poly')
svc.fit(X_train, y_train)
y_pred_test_svc = svc.predict(X_test)
y_pred_train_svc = svc.predict(X_train)
ac2 = accuracy_score(y_train, y_pred_train_svc), accuracy_score(y_test, y_pred_test_svc)
ac2
```

```
(0.85, 0.88)
```

```
param_grid = {'degree': [1,2,3,4,5,6,7,8,9,10,11,12,13,14,15], 'kernel':['poly']}
grid = GridSearchCV(SVC(), param_grid, verbose=2, scoring='accuracy')
grid.fit(X_train, y_train)
grid.best_params_
```

```
Fitting 5 folds for each of 15 candidates, totalling 75 fits
[CV] END ..............................degree=1, kernel=poly; total time=   0.0s
[CV] END ..............................degree=1, kernel=poly; total time=   0.0s
[CV] END ..............................degree=1, kernel=poly; total time=   0.0s
[CV] END ..............................degree=1, kernel=poly; total time=   0.0s
[CV] END ..............................degree=1, kernel=poly; total time=   0.0s
[CV] END ..............................degree=2, kernel=poly; total time=   0.0s
[CV] END ..............................degree=2, kernel=poly; total time=   0.0s
[CV] END ..............................degree=2, kernel=poly; total time=   0.0s
[CV] END ..............................degree=2, kernel=poly; total time=   0.0s
[CV] END ..............................degree=2, kernel=poly; total time=   0.0s
[CV] END ..............................degree=3, kernel=poly; total time=   0.0s
[CV] END ..............................degree=3, kernel=poly; total time=   0.0s
[CV] END ..............................degree=3, kernel=poly; total time=   0.0s
[CV] END ..............................degree=3, kernel=poly; total time=   0.0s
[CV] END ..............................degree=3, kernel=poly; total time=   0.0s
[CV] END ..............................degree=4, kernel=poly; total time=   0.0s
[CV] END ..............................degree=4, kernel=poly; total time=   0.0s
[CV] END ..............................degree=4, kernel=poly; total time=   0.0s
[CV] END ..............................degree=4, kernel=poly; total time=   0.0s
[CV] END ..............................degree=4, kernel=poly; total time=   0.0s
[CV] END ..............................degree=5, kernel=poly; total time=   0.0s
[CV] END ..............................degree=5, kernel=poly; total time=   0.0s
[CV] END ..............................degree=5, kernel=poly; total time=   0.0s
[CV] END ..............................degree=5, kernel=poly; total time=   0.0s
[CV] END ..............................degree=5, kernel=poly; total time=   0.0s
[CV] END ..............................degree=6, kernel=poly; total time=   0.0s
[CV] END ..............................degree=6, kernel=poly; total time=   0.0s
[CV] END ..............................degree=6, kernel=poly; total time=   0.0s
[CV] END ..............................degree=6, kernel=poly; total time=   0.0s
[CV] END ..............................degree=6, kernel=poly; total time=   0.0s
[CV] END ..............................degree=7, kernel=poly; total time=   0.0s
[CV] END ..............................degree=7, kernel=poly; total time=   0.0s
[CV] END ..............................degree=7, kernel=poly; total time=   0.0s
[CV] END ..............................degree=7, kernel=poly; total time=   0.0s
```

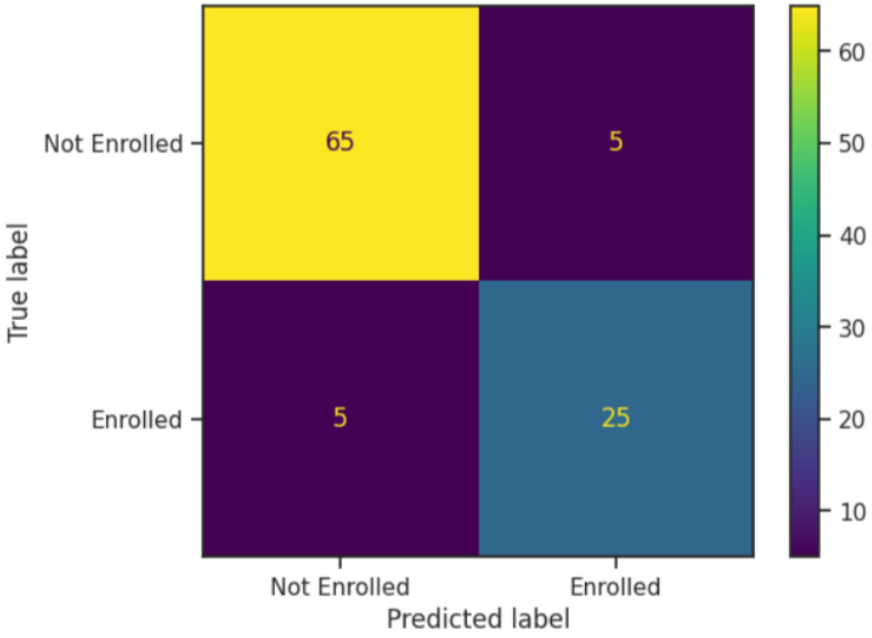
```
[CV] END ..............................degree=8, kernel=poly; total time=   0.0s
[CV] END ..............................degree=8, kernel=poly; total time=   0.0s
[CV] END ..............................degree=8, kernel=poly; total time=   0.0s
[CV] END ..............................degree=8, kernel=poly; total time=   0.0s
[CV] END ..............................degree=8, kernel=poly; total time=   0.0s
[CV] END ..............................degree=9, kernel=poly; total time=   0.0s
[CV] END ..............................degree=9, kernel=poly; total time=   0.0s
[CV] END ..............................degree=9, kernel=poly; total time=   0.0s
[CV] END ..............................degree=9, kernel=poly; total time=   0.0s
[CV] END ..............................degree=9, kernel=poly; total time=   0.0s
[CV] END .............................degree=10, kernel=poly; total time=   0.0s
[CV] END .............................degree=10, kernel=poly; total time=   0.0s
[CV] END .............................degree=10, kernel=poly; total time=   0.0s
[CV] END .............................degree=10, kernel=poly; total time=   0.0s
[CV] END .............................degree=10, kernel=poly; total time=   0.0s
[CV] END .............................degree=11, kernel=poly; total time=   0.0s
[CV] END .............................degree=11, kernel=poly; total time=   0.0s
[CV] END .............................degree=11, kernel=poly; total time=   0.0s
[CV] END .............................degree=11, kernel=poly; total time=   0.0s
[CV] END .............................degree=11, kernel=poly; total time=   0.0s
[CV] END .............................degree=12, kernel=poly; total time=   0.1s
[CV] END .............................degree=12, kernel=poly; total time=   0.0s
[CV] END .............................degree=12, kernel=poly; total time=   0.1s
[CV] END .............................degree=12, kernel=poly; total time=   0.0s
[CV] END .............................degree=12, kernel=poly; total time=   0.0s
[CV] END .............................degree=13, kernel=poly; total time=   0.2s
[CV] END .............................degree=13, kernel=poly; total time=   0.1s
[CV] END .............................degree=13, kernel=poly; total time=   0.2s
[CV] END .............................degree=13, kernel=poly; total time=   0.2s
[CV] END .............................degree=13, kernel=poly; total time=   0.2s
[CV] END .............................degree=14, kernel=poly; total time=   1.2s
[CV] END .............................degree=14, kernel=poly; total time=   0.3s
[CV] END .............................degree=14, kernel=poly; total time=   0.7s
[CV] END .............................degree=14, kernel=poly; total time=   0.5s
[CV] END .............................degree=14, kernel=poly; total time=   0.4s
[CV] END .............................degree=15, kernel=poly; total time=   2.0s
[CV] END .............................degree=15, kernel=poly; total time=   0.5s
[CV] END .............................degree=15, kernel=poly; total time=   0.4s
[CV] END .............................degree=15, kernel=poly; total time=   0.6s
[CV] END .............................degree=15, kernel=poly; total time=   0.8s
{'degree': 14, 'kernel': 'poly'}
```

[12]
```
svc = SVC(kernel='poly', degree=14)
svc.fit(X_train, y_train)
y_pred_test_svc = svc.predict(X_test)
y_pred_train_svc = svc.predict(X_train)
accuracy_score(y_train, y_pred_train_svc), accuracy_score(y_test, y_pred_test_svc)
```

```
(0.905, 0.9)
```

```
[13] cm2 = confusion_matrix(y_test, y_pred_test_svc, labels = svc.classes_)
     disp = ConfusionMatrixDisplay(confusion_matrix=cm2, display_labels=['Not Enrolled', 'Enrolled'])
     disp.plot()
```

```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x7fdd77ffd060>
```

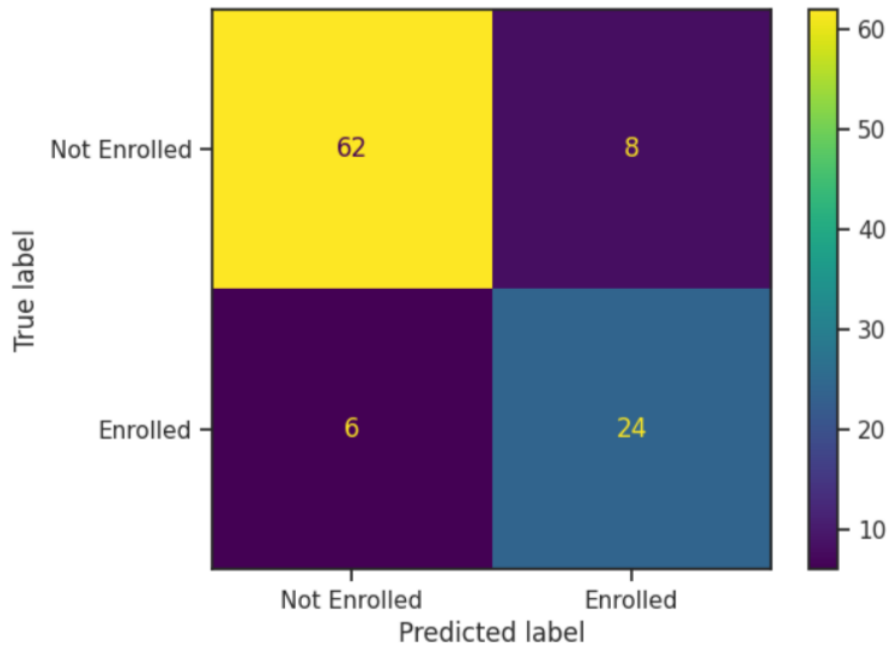



```
[14] tree = DecisionTreeClassifier(random_state=1).fit(X_train, y_train)
     y_pred_test_tree = tree.predict(X_test)
     y_pred_train_tree = tree.predict(X_train)
     ac3 = accuracy_score(y_train, y_pred_train_tree), accuracy_score(y_test, y_pred_test_tree)
     ac3
```

```
(1.0, 0.86)
```

```
[15] cm3 = confusion_matrix(y_test, y_pred_test_tree, labels = tree.classes_)
     disp = ConfusionMatrixDisplay(confusion_matrix=cm3, display_labels=['Not Enrolled', 'Enrolled'])
     disp.plot()
```

```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x7fdd77e77d90>
```



```
[16] # сравнение качества моделей по 2 метрикам
     print('LogisticRegression:     ', ac1)
     print('SVM:                    ', ac2)
     print('DecisionTreeClassifier: ', ac3)
```

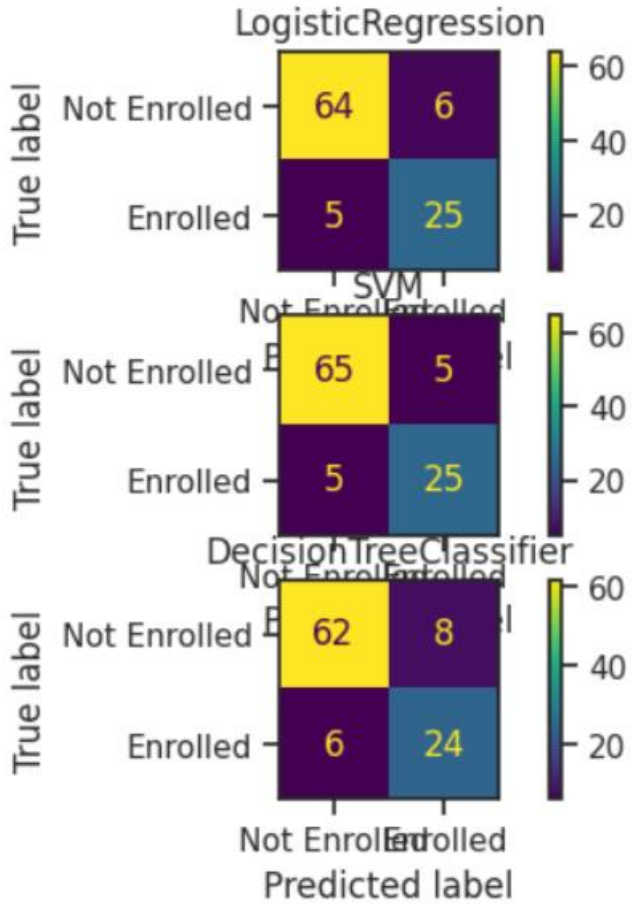
```
LogisticRegression:      (0.9, 0.89)
SVM:                     (0.85, 0.88)
DecisionTreeClassifier:  (1.0, 0.86)
```

```
[17] fig, ax = plt.subplots(3,1)
     ax[0].set_title("LogisticRegression")
     ax[1].set_title("SVM")
     ax[2].set_title("DecisionTreeClassifier")

     ConfusionMatrixDisplay(confusion_matrix=cm1, display_labels=['Not Enrolled', 'Enrolled']).plot(ax=ax[0])

     ConfusionMatrixDisplay(confusion_matrix=cm2, display_labels=['Not Enrolled', 'Enrolled']).plot(ax=ax[1])

     ConfusionMatrixDisplay(confusion_matrix=cm3, display_labels=['Not Enrolled', 'Enrolled']).plot(ax=ax[2])
```



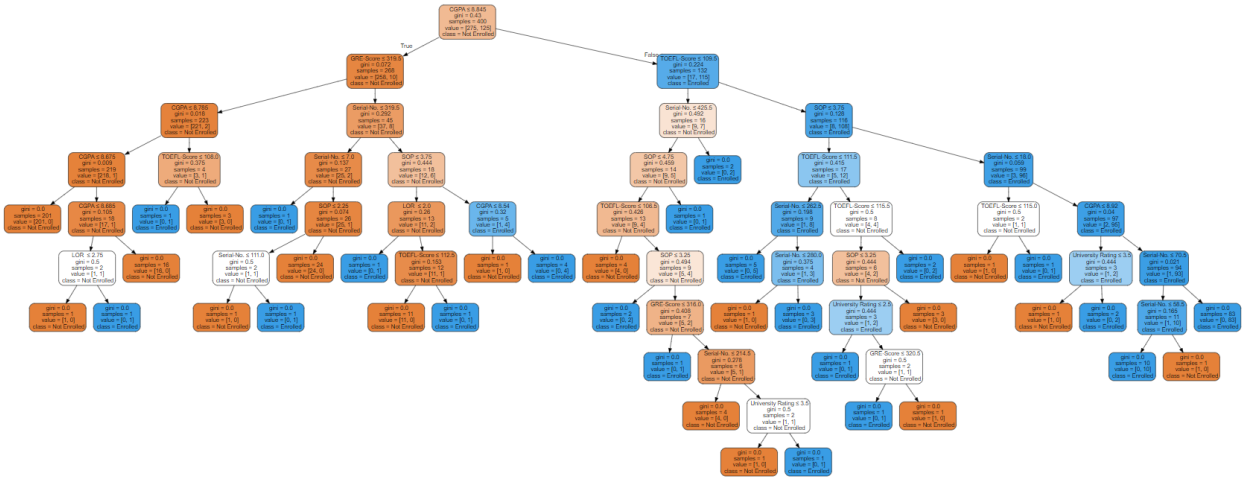

```
tree_rules = export_text(tree, feature_names=list(dataset.iloc[:, :-1].columns))
HTML('<pre>' + tree_rules + '</pre>')
```

```
|--- CGPA <= 8.85
|   |--- GRE-Score <= 319.50
|   |   |--- CGPA <= 8.78
|   |   |   |--- CGPA <= 8.68
|   |   |   |   |--- class: 0
|   |   |   |--- CGPA >  8.68
|   |   |   |   |--- CGPA <= 8.68
```

```
|   |   |   |   |   |--- LOR  <= 2.75
|   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |--- LOR  >  2.75
|   |   |   |   |   |   |--- class: 1
|   |   |   |   |--- CGPA >  8.68
|   |   |   |   |   |--- class: 0
|   |   |--- CGPA >  8.78
|   |   |   |--- TOEFL-Score <= 108.00
|   |   |   |   |--- class: 1
|   |   |   |--- TOEFL-Score >  108.00
|   |   |   |   |--- class: 0
|   |--- GRE-Score >  319.50
|   |   |--- Serial-No. <= 319.50
|   |   |   |--- Serial-No. <= 7.00
|   |   |   |   |--- class: 1
|   |   |   |--- Serial-No. >  7.00
|   |   |   |   |--- SOP <= 2.25
|   |   |   |   |   |--- Serial-No. <= 111.00
|   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |--- Serial-No. >  111.00
|   |   |   |   |   |   |--- class: 1
|   |   |   |   |--- SOP >  2.25
|   |   |   |   |   |--- class: 0
|   |   |--- Serial-No. >  319.50
|   |   |   |--- SOP <= 3.75
|   |   |   |   |--- LOR  <= 2.00
|   |   |   |   |   |--- class: 1
|   |   |   |   |--- LOR  >  2.00
|   |   |   |   |   |--- TOEFL-Score <= 112.50
|   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |--- TOEFL-Score >  112.50
|   |   |   |   |   |   |--- class: 1
|   |   |   |--- SOP >  3.75
|   |   |   |   |--- CGPA <= 8.54
|   |   |   |   |   |--- class: 0
|   |   |   |   |--- CGPA >  8.54
|   |   |   |   |   |--- class: 1
|--- CGPA >  8.85
|   |--- TOEFL-Score <= 109.50
|   |   |--- Serial-No. <= 425.50
|   |   |   |--- SOP <= 4.75
|   |   |   |   |--- TOEFL-Score <= 106.50
|   |   |   |   |   |--- class: 0
|   |   |   |   |--- TOEFL-Score >  106.50
|   |   |   |   |   |--- SOP <= 3.25
|   |   |   |   |   |   |--- class: 1
|   |   |   |   |   |--- SOP >  3.25
|   |   |   |   |   |   |--- GRE-Score <= 316.00
|   |   |   |   |   |   |   |--- class: 1
|   |   |   |   |   |   |--- GRE-Score >  316.00
|   |   |   |   |   |   |   |--- Serial-No. <= 214.50
|   |   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |   |   |--- Serial-No. >  214.50
|   |   |   |   |   |   |   |   |--- University Rating <= 3.50
|   |   |   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |   |   |   |--- University Rating >  3.50
|   |   |   |   |   |   |   |   |   |--- class: 1
|   |   |   |--- SOP >  4.75
|   |   |   |   |--- class: 1
|   |   |--- Serial-No. >  425.50
|   |   |   |--- class: 1
```

```
|   |--- TOEFL-Score >  109.50
|   |   |--- SOP <= 3.75
|   |   |   |--- TOEFL-Score <= 111.50
|   |   |   |   |--- Serial-No. <= 262.50
|   |   |   |   |   |--- class: 1
|   |   |   |   |--- Serial-No. >  262.50
|   |   |   |   |   |--- Serial-No. <= 280.00
|   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |--- Serial-No. >  280.00
|   |   |   |   |   |   |--- class: 1
|   |   |   |--- TOEFL-Score >  111.50
|   |   |   |   |--- TOEFL-Score <= 115.50
|   |   |   |   |   |--- SOP <= 3.25
|   |   |   |   |   |   |--- University Rating <= 2.50
|   |   |   |   |   |   |   |--- class: 1
|   |   |   |   |   |   |--- University Rating >  2.50
|   |   |   |   |   |   |   |--- GRE-Score <= 320.50
|   |   |   |   |   |   |   |   |--- class: 1
|   |   |   |   |   |   |   |--- GRE-Score >  320.50
|   |   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |--- SOP >  3.25
|   |   |   |   |   |   |--- class: 0
|   |   |   |   |--- TOEFL-Score >  115.50
|   |   |   |   |   |--- class: 1
|   |   |--- SOP >  3.75
|   |   |   |--- Serial-No. <= 18.00
|   |   |   |   |--- TOEFL-Score <= 115.00
|   |   |   |   |   |--- class: 0
|   |   |   |   |--- TOEFL-Score >  115.00
|   |   |   |   |   |--- class: 1
|   |   |   |--- Serial-No. >  18.00
|   |   |   |   |--- CGPA <= 8.92
|   |   |   |   |   |--- University Rating <= 3.50
|   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |--- University Rating >  3.50
|   |   |   |   |   |   |--- class: 1
|   |   |   |   |--- CGPA >  8.92
|   |   |   |   |   |--- Serial-No. <= 70.50
|   |   |   |   |   |   |--- Serial-No. <= 58.50
|   |   |   |   |   |   |   |--- class: 1
|   |   |   |   |   |   |--- Serial-No. >  58.50
|   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |--- Serial-No. >  70.50
|   |   |   |   |   |   |--- class: 1
```

```
dot_data = export_graphviz(tree, out_file=None,
                           feature_names=dataset.iloc[:, :-1].columns,
                           class_names=['Not Enrolled', 'Enrolled'],
                           filled=True, rounded=True, special_characters=True)
graph = graphviz.Source(dot_data)
graph
```



```python
from operator import itemgetter

def draw_feature_importances(tree_model, X_dataset, figsize=(18,5)):
    """
    Вывод важности признаков в виде графика
    """
    # Сортировка значений важности признаков по убыванию
    list_to_sort = list(zip(X_dataset.columns.values, tree_model.feature_importances_))
    sorted_list = sorted(list_to_sort, key=itemgetter(1), reverse = True)
    # Названия признаков
    labels = [x for x,_ in sorted_list]
    # Важности признаков
    data = [x for _,x in sorted_list]
    # Вывод графика
    fig, ax = plt.subplots(figsize=figsize)
    ind = np.arange(len(labels))
    plt.bar(ind, data)
    plt.xticks(ind, labels, rotation='vertical')
    # Вывод значений
    for a,b in zip(ind, data):
        plt.text(a-0.05, b+0.01, str(round(b,3)))
    plt.show()
    return labels, data
```

```
[ ]  fl, fd = draw_feature_importances(tree, dataset.iloc[:, :-1])
```