Ордена Трудового Красного Знамени федеральное государственное бюджетное образовательное учреждение высшего образования «Московский технический университет связи и информатики»

Кафедра МКиИТ

Лабораторная работа №1 по дисциплине

«Программное обеспечение систем искусственного интеллекта»

Группа: М091901(76)  
Выполнил: Денисов В.В.

Москва 2021

**Задание:** необходимо взять датасет, относящийся к одной из категорий:

1. Датасет, используемый в магистерской диссертации
2. Датасет с работы
3. Датасет с площадок для соревнований по машинному обучению (Например kaggle)
4. Сгенерированный датасет (Например, спарсенные картинки и т.д.)

Далее на имеющемся датасете обучается несколько моделей и замеряются метрики качества.

**Результаты выполнения лабораторной работы:**

В результате выполнения лабораторной работы была написана компьютерная программа на языке Python, листинг которой представлен в приложениях 1 и 2, а также выложен в репозиторий по ссылке:

<https://github.com/VaskaKasatka/DenisovLabPosii>

Программа выполняет бинарную классификацию индикатора счастья в зависимости от континента, в котором живет человек. Работает с помощью нескольких алгоритмов машинного обучения. В конце программа выводит различные метрики, характеризующие эффективность работы моделей, а наилучшую по метрикам модель - сохраняет на диск.

Основные файлы:

1. main.py - код, выполняющий обучение и тестирование моделей.
2. read\_dataset.py - код, выполняющий чтение датасета из файла "2019.csv".

Были обучены и протестированы следующие модели:

* модель с методом k-ближайших соседей;
* модель с методом опорных векторов;
* модель дерева решений;
* модель многослойного персептрона.

Результат работы программы можно наблюдать на рисунке 1.

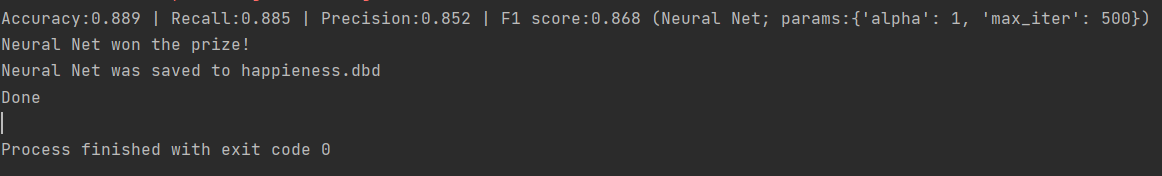


Рисунок 1 - Эффективность моделей

**Выводы по лабораторной работе:** написал компьютерную программу на языке Python, создающую, обучающую и тестирующую несколько моделей классификации; изучил основы работы использованных моделей; изучил основы оценивания моделей классификации.

**Приложение 1**

from sklearn.model\_selection import train\_test\_split, GridSearchCV  
from sklearn.preprocessing import StandardScaler  
from sklearn.metrics import accuracy\_score, precision\_score, recall\_score, f1\_score  
from sklearn.neural\_network import MLPClassifier  
from sklearn.neighbors import KNeighborsClassifier  
from sklearn.svm import SVC  
from sklearn.tree import DecisionTreeClassifier  
from pickle import dump  
  
import read\_dataset  
  
names = ["Nearest Neighbors", "Linear SVM", "Decision Tree", "Neural Net"]  
scoring = ["accuracy", "recall", "precision", "f1"]  
  
kncParams = {'n\_neighbors': [3, 5, 10]}  
svcParams = {'C': [0.025, 0.5, 1]}  
dtcParams = {'max\_depth': [5, 10, 15]}  
mlpParams = {'alpha': [0.0001, 0.001, 1], 'max\_iter': [500, 1000, 1500]}  
tuneParams = [kncParams, svcParams, dtcParams, mlpParams]  
  
classifiers = [  
 KNeighborsClassifier(),  
 SVC(kernel="linear"),  
 DecisionTreeClassifier(),  
 MLPClassifier()]  
  
X, y = read\_dataset.get\_happieness\_dataset()  
  
X = StandardScaler().fit\_transform(X)  
y = y.astype('int')  
  
X\_train, X\_test, y\_train, y\_test = \  
 train\_test\_split(X, y, test\_size=.4, random\_state=42)  
  
wholeBestEstimator = None  
wholeBestScore = 0.0  
  
# iterate over classifiers  
for name, clf, params in zip(names, classifiers, tuneParams):  
 gsCV = GridSearchCV(clf, params, scoring=scoring, refit='accuracy')  
 gsCV.fit(X\_train, y\_train)  
 if wholeBestScore < gsCV.best\_score\_:  
 wholeBestEstimator = (name, gsCV.best\_estimator\_)  
 wholeBestScore = gsCV.best\_score\_  
  
 y\_pred = gsCV.predict(X\_test)  
 accuracy = accuracy\_score(y\_test, y\_pred)  
 recall = recall\_score(y\_test, y\_pred)  
 precision = precision\_score(y\_test, y\_pred)  
 f1 = f1\_score(y\_test, y\_pred)  
 print("Accuracy:{:.3f}".format(accuracy),  
 "| Recall:{:.3f}".format(recall),  
 "| Precision:{:.3f}".format(precision),  
 "| F1 score:{:.3f}".format(f1),  
 "({0}; params:{1})".format(name, gsCV.best\_params\_))  
  
print(wholeBestEstimator[0] + " won the prize!")  
  
bestFileNameEver = "happieness.dbd"  
dump(wholeBestEstimator, open(bestFileNameEver, 'wb'))  
  
print(wholeBestEstimator[0] + " was saved to " + bestFileNameEver)  
print("Done")

**Приложение 2**

import pandas as pd  
  
  
def get\_happieness\_dataset():  
 df = pd.read\_csv(r'2019.csv')  
 return df.drop('happieness', axis=1), df['happieness']