MINISTRY OF SCIENCE AND EDUCATION OF UKRAINE

TARAS SHEVCHENKA NATIONAL UNIVERSITY OF KYIV

INFORMATION TECHNOLOGIES FACULTY

LABORATORY WORK № 1 REPORT BY TOPIC:

**CLASSIFICATION**

|  |  |
| --- | --- |
| Group | \_\_\_\_\_\_\_\_11\_\_\_\_\_\_\_\_\_\_\_ |
| Course | \_\_\_\_\_\_\_\_\_1\_\_\_\_\_\_\_\_\_\_\_ |
| Student | \_\_\_Maxim Suprunenko\_\_ |
| Data | \_\_\_\_\_\_21.03.2024\_\_\_\_\_\_ |
| Checked by | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |
| Data | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |

THE GOAL OF THE WORK: The purpose of the work is to familiarize students with the basics approaches in different type of classification.

THEORY: Theoretical and additional materials on the topics of laboratory work are presented in detail in the materials of lectures 2, as well as during practical work No. 1 (part \_\_).

PROGRESS:

Explore the Titanic dataset. A good place to start is Kaggle (https://www. kaggle.com/c/titanic).

The goal is to predict whether or not a passenger survived based on attributes such as their age, sex, passenger class, where they embarked and so on.

First, login to [Kaggle](https://www.kaggle.com/) and go to the Titanic challenge to download train.csv and test.csv. Save them to the datasets/titanic (or your) directory.

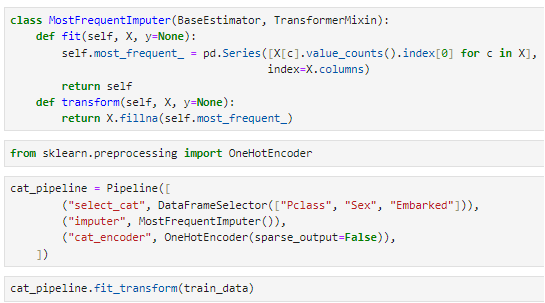
The data is already split into a training set and a test set. However, the test data does not contain the labels: your goal is to train the best model you can using the training data, then make your predictions on the test data and upload them to Kaggle to see your final score.

The attributes have the following meaning:

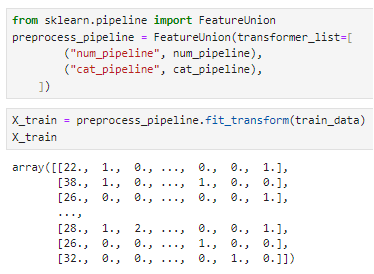
* **Survived**: that's the target, 0 means the passenger did not survive, while 1 means he/she survived.
* **Pclass**: passenger class.
* **Name**, **Sex**, **Age**: self-explanatory
* **SibSp**: how many siblings & spouses of the passenger aboard the Titanic.
* **Parch**: how many children & parents of the passenger aboard the Titanic.
* **Ticket**: ticket id
* **Fare**: price paid (in pounds)
* **Cabin**: passenger's cabin number
* **Embarked**: where the passenger embarked the Titanic (C=Cherbourg, Q=Queenstown, S=Southampton.)
  1. Explore the training dataset in detail (train\_data.head(), train\_data.info()), make a conclusion.
  2. Take a look at the numerical attributes: train\_data**.**describe()
  3. Check that the target is indeed 0 or 1: train\_data["Survived"]**.**value\_counts()
  4. Take a quick look at all the categorical attributes: train\_data["Name"]**.**value\_counts()
  5. Build preprocessing pipelines for the numerical attributes



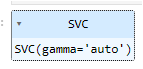
* 1. Build the pipeline for the categorical attributes



* 1. Join the numerical and categorical pipelines (**from** sklearn.pipeline **import** FeatureUnion).



* 1. We are now ready to train a classifier. Let's start with an SVC:



* 1. The model is trained; use it to make predictions on the test set.
  2. Build a confusion matrix, make a conclusion.
  3. Use cross-validation to have an idea of how good your model is. Divide train set into 10 subsets (cv=10). Make a conclusion; try to build a model that reaches 80% accuracy.
  4. Try a RandomForestClassifier. Build a confusion matrix, make a conclusion.
  5. Plot all 10 scores for each model, along with a box plot highlighting the lower and upper quartiles, and "whiskers" showing the extent of the scores. Note that the boxplot() function detects outliers (called "fliers") and does not include them within the whiskers. Specifically, if the lower quartile is Q1 and the upper quartile is Q3, then the interquartile range IQR=Q3−Q1 (this is the box's height), and any score lower than Q1−1.5×IQR is a flier, and so is any score greater than Q3+1.5×IQR.

**LABORATORY WORK № 1**

**1. Import library**

|  |
| --- |
| import pandas as pd  import numpy as np  import matplotlib.pyplot as plt  import seaborn as sns  from sklearn.model\_selection import train\_test\_split, cross\_val\_score  from sklearn.preprocessing import StandardScaler, OneHotEncoder  from sklearn.impute import SimpleImputer  from sklearn.pipeline import Pipeline  from sklearn.compose import ColumnTransformer  from sklearn.svm import SVC  from sklearn.ensemble import RandomForestClassifier  from sklearn.metrics import accuracy\_score, confusion\_matrix, classification\_report |

1. **Install date**

|  |
| --- |
| train\_data = pd.read\_csv("train.csv")  test\_data = pd.read\_csv("test.csv") |

1. **Check Data structure**

|  |
| --- |
| train\_data.head()    train\_data.info()    train\_data.describe() |

1. **Explore null**

|  |
| --- |
| train\_data.isnull().sum() |

1. **Correlation**

|  |
| --- |
| train\_data.select\_dtypes(include=['number']).corr() |
|  |
| numeric\_data = train\_data.select\_dtypes(include=['number'])  plt.figure(figsize=(10, 6))  sns.heatmap(numeric\_data.corr(), annot=True, cmap="coolwarm", fmt=".2f")  plt.title("Кореляційна матриця")  plt.show() |
|  |
| Pclass (-0.338) – A negative correlation indicates that passengers in lower classes (Pclass = 3) had a lower chance of survival.  Fare (0.257) – A positive correlation suggests that passengers with more expensive tickets had a higher chance of survival.  Pclass and Fare (-0.549) – Passengers in higher classes (1st class) paid significantly more for their tickets.  SibSp and Parch (0.414) – Logically, passengers traveling with family (children and parents) were also often accompanied by siblings or spouses. |

1. **Check value survived**

|  |
| --- |
| train\_data["Survived"].value\_counts() |
|  |

1. **Scale and Impute**

|  |
| --- |
| num\_features = ["Age", "Fare", "Parch", "SibSp"]  num\_pipeline = Pipeline([      ("imputer", SimpleImputer(strategy="median")),      ("scaler", StandardScaler()),  ]) |
| cat\_features = ["Pclass", "Sex", "Embarked"]  cat\_pipeline = Pipeline([      ("imputer", SimpleImputer(strategy="most\_frequent")),      ("encoder", OneHotEncoder(handle\_unknown="ignore")),  ]) |
| preprocessor = ColumnTransformer([      ("num", num\_pipeline, num\_features),      ("cat", cat\_pipeline, cat\_features),  ]) |

1. **Data split**

|  |
| --- |
| X = train\_data.drop(["Survived", "Name", "Ticket", "Cabin"], axis=1)  y = train\_data["Survived"]  X\_train, X\_valid, y\_train, y\_valid = train\_test\_split(X, y, test\_size=0.2, random\_state=42) |

1. **SVC model**

|  |
| --- |
| svc\_model = Pipeline([      ("preprocessor", preprocessor),      ("classifier", SVC(kernel="linear", C=1))  ])  svc\_model.fit(X\_train, y\_train)  y\_pred\_svc = svc\_model.predict(X\_valid) |
| print("Accuracy:", accuracy\_score(y\_valid, y\_pred\_svc))  print(confusion\_matrix(y\_valid, y\_pred\_svc))  print(classification\_report(y\_valid, y\_pred\_svc)) |
|  |
| svc\_scores = cross\_val\_score(svc\_model, X\_train, y\_train, cv=10, scoring="accuracy")  print("SVC mean accuracy:", svc\_scores.mean()) |
|  |

1. **Random forest**

|  |
| --- |
| rf\_model = Pipeline([      ("preprocessor", preprocessor),      ("classifier", RandomForestClassifier(n\_estimators=200, random\_state=42))  ])  rf\_model.fit(X\_train, y\_train)  y\_pred\_rf = rf\_model.predict(X\_valid) |
| print("Accuracy:", accuracy\_score(y\_valid, y\_pred\_rf))  print(confusion\_matrix(y\_valid, y\_pred\_rf))  print(classification\_report(y\_valid, y\_pred\_rf)) |
|  |

1. **Model comparison**

|  |
| --- |
| plt.figure(figsize=(8, 5))  plt.boxplot([svc\_scores, rf\_scores], labels=["SVC", "RandomForest"])  plt.ylabel("Accuracy")  plt.title("Model Comparison")  plt.show() |
|  |