def getConfigFile(config):

with open(config, encoding='utf-8') as json\_file:

return json.load(json\_file)

def addTextParagraphToDocumentInStyle(text, document, style):

p = document.add\_paragraph(text)

p.style = document.styles[style]

def AddBookmark(t, mark):

# Find a specific text or phrase in the document

paragraph = section.AddParagraph()

paragraph.Format.HorizontalAlignment = HorizontalAlignment.Center

text = paragraph.AppendText(t)

text = document.FindString(t, False, True)

# Get the found text as a single text range

textRange = text.GetAsOneRange()

# Get the paragraph where the text range is located

paragraph = textRange.OwnerParagraph

# Get the index position of the text in the paragraph

index = paragraph.ChildObjects.IndexOf(textRange)

# Add a bookmark start mark to the paragraph

start = paragraph.AppendBookmarkStart(mark)

# Insert the bookmark start mark at the index position of the text range

paragraph.ChildObjects.Insert(index, start)

# Add a bookmark end mark to the paragraph

end = paragraph.AppendBookmarkEnd(mark)

# Insert the bookmark end mark after the text range

paragraph.ChildObjects.Insert(index + 2, end)

def update\_toc(docx\_file):

word = win32com.client.DispatchEx("Word.Application")

word.Visible = 1

word.DisplayAlerts = 0

doc = word.Documents.Open(docx\_file)

#wd\_section = doc.Sections(1)

toc\_count = doc.TablesOfContents.Count

print(toc\_count)

stringG='INHALTSVERZEICHNIS'

stringK='Содержание'

if toc\_count == 0:

for i, p in enumerate(doc.Paragraphs):

if stringK in p.Range.Text:

try:

p.Range.InsertParagraphAfter()

parag\_range = doc.Paragraphs(i+2).Range

parag\_range.Font.Name = 'Arial'

parag\_range.Font.Size = 14

parag\_range.Font.Bold = constants.wdToggle

parag\_range.Font.Size = 12

doc.TablesOfContents.Add(Range=parag\_range,

UseHeadingStyles=True,

LowerHeadingLevel=3)

except Exception as e:

print("Ja：", e, "Nein")

break

elif toc\_count == 1:

toc = doc.TablesOfContents(1)

toc.Update()

print('TOC should have been updated.')

else:

print('TOC has not been updated for sure...')

doc.Close()

word.Quit()

def set\_column\_width(table, column, width\_mm):

table.allow\_autofit = False

for row in table.rows:

row.cells[column].width = Mm(width\_mm)

def set\_repeat\_table\_header(row):

tr = row.\_tr

trPr = tr.get\_or\_add\_trPr()

tblHeader = OxmlElement('w:tblHeader')

tblHeader.set(qn('w:val'), "true")

trPr.append(tblHeader)

return row

def change\_table\_cell(cell, background\_color=None, font\_color=None, font\_size=None, bold=None, italic=None):

if background\_color:

shading\_elm = parse\_xml(r'<w:shd {} w:fill="{}"/>'.format(nsdecls('w'), background\_color))

cell.\_tc.get\_or\_add\_tcPr().append(shading\_elm)

if font\_color:

for p in cell.paragraphs:

for r in p.runs:

r.font.color.rgb = docx.shared.RGBColor.from\_string(font\_color)

if font\_size:

for p in cell.paragraphs:

for r in p.runs:

r.font.size = docx.shared.Pt(font\_size)

if bold is not None:

for p in cell.paragraphs:

for r in p.runs:

r.bold = bold

if italic is not None:

for p in cell.paragraphs:

for r in p.runs:

r.italic = italic

def set\_cell\_border(cell, \*\*kwargs):

tc = cell.\_tc

tcPr = tc.get\_or\_add\_tcPr()

# check for tag existnace, if none found, then create one

tcBorders = tcPr.first\_child\_found\_in("w:tcBorders")

if tcBorders is None:

tcBorders = OxmlElement('w:tcBorders')

tcPr.append(tcBorders)

for edge in ('left', 'top', 'right', 'bottom', 'insideH', 'insideV'):

edge\_data = kwargs.get(edge)

if edge\_data:

tag = 'w:{}'.format(edge)

# check for tag existnace, if none found, then create one

element = tcBorders.find(qn(tag))

if element is None:

element = OxmlElement(tag)

tcBorders.append(element)

# looks like order of attributes is important

for key in ["sz", "val", "color", "space", "shadow"]:

if key in edge\_data:

element.set(qn('w:{}'.format(key)), str(edge\_data[key]))

def delete\_paragraph(paragraph):

p = paragraph.\_element

p.getparent().remove(p)

p.\_p = p.\_element = None

def replace\_copy(file, txt):

wordapp = win32com.client.gencache.EnsureDispatch("Word.Application")

wordapp.Visible = True

newdoc = wordapp.Documents.Open(file)

#print(newdoc.Paragraphs.Count)

finder = wordapp.Selection.Find

finder.Text = txt #"Hier"

finder.Execute()

#wordapp.Selection.MoveLeft()

#wordapp.Selection.MoveDown()

wordapp.Selection.MoveStart

wordapp.Selection.Paste()

newdoc.SaveAs("C:\IPYNBgesamt\ASNI-FEN\ASNI-Report\ASNI\_ReportResult.docx")

newdoc .ActiveWindow.Close()

wordapp.Application.Quit(-1)

def Text\_copy(file):

wordapp = win32com.client.gencache.EnsureDispatch("Word.Application")

wordapp.Visible = True

worddoc = wordapp.Documents.Open(file)

worddoc.Select()

wordapp.Selection.Copy()

worddoc.ActiveWindow.Close()

wordapp.Application.Quit(-1)

# In[3]:

doc = DocxTemplate("rTemplateTest.docx")

reportWordPath = 'ASNI\_ReportTest01.docx'

Inmodels = pd.DataFrame({'Model': [

'Linear Regression',

'Logistic Regression',

'Perceptron',

'Linear SVC',

'MLPClassifier',

'Decision Tree Classifier 1',

'Stochastic Gradient Decent',

'RidgeClassifier',

'BaggingClassifier',

'AdaBoostClassifier 1',

'GradientBoostingClassifie',

'KNeighborsClassifier',

'DecisionTreeClassifier 2',

'RandomForestClassifier',

'XGBClassifier',

'AdaBoostClassifier 2',

'Naive Bayes',

'SVC' ]})

Nk=len(Inmodels)

print(Nk)

Asni = {

'Projekt': 'Asfendiyarov Kazakh National Medical University',

'Projekt2': 'Statistik und Data Science Projekt',

'Projekt3': 'Практическое применение Автоматизированной системы научных исследований в медицине, здравоохранении и смежных областях',

'Thema': 'Анализ факторов риска сердечно сосудистых заболеваний и прогноз исходов лечения при помощи методов Машинного Обучения',

'Forscher': 'Dr. Alexander Wagner (Berlin)',

'Site' : 'Berlin-Almaty',

'Year' : str(year),

"tasks" : [

{

"folder" : "0",

"include" : ["Book", "Dog"],

"topic" : "Содержание",

"models" : " ",

"Text" : " "

},

{

"folder" : "0",

"include" : ["Book", "Dog"],

"topic" : "Предисловие",

"models" : "xHier",

"Text" : "File"

},

{

"folder" : "1",

"include" : ["Book", "Dog"],

"topic" : "Введение",

"models" : " ",

"Text" : "Kap01"

},

{

"folder" : "2",

"include" : ["Author", "Ball"],

"topic" : "Цель исследования",

"models" : " ",

"Text" : "Kap02"

},

{

"folder" : "3",

"include" : ["Author", "Ball"],

"topic" : "Материалы и методы",

"models" : " ",

"Text" : "Kap03"

},

{

"folder" : "4",

"include" : ["Author", "Ball"],

"topic" : "Исходные данные и их организация",

"models" : " ",

"Text" : "Kap04"

},

{

"folder" : "5",

"include" : ["Author", "Ball"],

"topic" : "Предварительный анализ данных",

"models" : " ",

"Text" : "Kap05"

},

{

"folder" : "6",

"include" : ["MovablePoint", "Rectangle"],

"topic" : "Моделирование",

"models" : [

'Linear Regression',

'Logistic Regression',

'Perceptron',

'Linear SVC',

'MLPClassifier',

'Decision Tree Classifier 1',

'Stochastic Gradient Decent',

'RidgeClassifier',

'BaggingClassifier',

'AdaBoostClassifier 1',

'GradientBoostingClassifie',

'KNeighborsClassifier',

'DecisionTreeClassifier 2',

'RandomForestClassifier',

'XGBClassifier',

'AdaBoostClassifier 2',

'Naive Bayes',

'SVC' ],

"Text" : "Kap06"

},

{

"folder" : "7",

"include" : ["Exercise", "TextAreaExample"],

"topic" : "Результаты моделирования",

"models" : " ",

"Text" : "Kap07"

},

{

"folder" : "8",

"include" : ["Exercise", "TextAreaExample"],

"topic" : "Оценка моделей и рекомендации",

"models" : " ",

"Text" : "Kap08"

},

{

"folder" : "9",

"include" : ["Exercise", "TextAreaExample"],

"topic" : "Обсуждение и выводы",

"models" : " ",

"Text" : "Kap09"

},

{

"folder" : "10",

"include" : ["Exercise", "TextAreaExample"],

"topic" : "Заключение",

"models" : " ",

"Text" : "Kap010"

},

{

"folder" : "11",

"include" : ["Exercise", "TextAreaExample"],

"topic" : "Литература",

"models" : "xLiter",

"Text" : "File"

},

],

}

with open("Templates/ASNIR.json", "w", encoding="utf-8") as file\_handle:

json.dump(Asni, file\_handle, indent=4)

# In[4]:

#Spire

from spire.doc.common import \*

from spire.doc import \*

CONFIG\_JSON = "Templates/ASNIR.json"

# Используемые стили

HEADER\_STYLE = "BoldHeader"

HEADER\_LINK\_STYLE = "BoldHeaderHyperlink"

CONTENT\_STYLE = "Content"

CODE\_STYLE = "Code"

document = Document()

section = document.AddSection()

paragraph = section.AddParagraph()

paragraph.Format.HorizontalAlignment = HorizontalAlignment.Center

text =paragraph.AppendText("{{Projekt}}")

text.CharacterFormat.FontName = "Times New Roman"

text.CharacterFormat.FontSize = 26

text.CharacterFormat.Bold = True

text.CharacterFormat.TextColor = Color.get\_Blue()

paragraph = section.AddParagraph()

paragraph = section.AddParagraph()

paragraph.Format.HorizontalAlignment = HorizontalAlignment.Center

text = paragraph.AppendText("Тема исследования: {{Projekt3}}")

text.CharacterFormat.FontName = "Times New Roman"

text.CharacterFormat.FontSize = 16

text.CharacterFormat.Bold = True

paragraph = section.AddParagraph()

paragraph = section.AddParagraph()

paragraph.Format.HorizontalAlignment = HorizontalAlignment.Center

paragraph.Format.HorizontalAlignment = HorizontalAlignment.Center

text = paragraph.AppendText("Проект: {{Thema}}")

text.CharacterFormat.FontName = "Times New Roman"

text.CharacterFormat.FontSize = 16

text.CharacterFormat.Bold = True

paragraph = section.AddParagraph()

paragraph = section.AddParagraph()

paragraph.Format.HorizontalAlignment = HorizontalAlignment.Center

text = paragraph.AppendText("Автор исследования: {{Forscher}}")

text.CharacterFormat.FontName = "Times New Roman"

text.CharacterFormat.FontSize = 16

text.CharacterFormat.Bold = True

for num in range(9):

i=num+1

paragraph = section.AddParagraph()

paragraph = section.AddParagraph()

paragraph = section.AddParagraph()

paragraph.Format.HorizontalAlignment = HorizontalAlignment.Center

text = paragraph.AppendText("{{logo}}")

text.CharacterFormat.FontName = "Times New Roman"

text.CharacterFormat.FontSize = 16

text.CharacterFormat.Bold = True

for num in range(13):

i=num+1

paragraph = section.AddParagraph()

paragraph = section.AddParagraph()

paragraph.Format.HorizontalAlignment = HorizontalAlignment.Center

text = paragraph.AppendText("{{Site}}")

text.CharacterFormat.FontName = "Times New Roman"

text.CharacterFormat.FontSize = 14

text.CharacterFormat.Bold = True

paragraph = section.AddParagraph()

paragraph.Format.HorizontalAlignment = HorizontalAlignment.Center

text = paragraph.AppendText("{{Year}}")

text.CharacterFormat.FontName = "Times New Roman"

text.CharacterFormat.FontSize = 14

text.CharacterFormat.Bold = True

outputFile = "TemplateTOC1.docx"

document.SaveToFile(outputFile, FileFormat.Docx)

document.Close()

# In[5]:

# Используемые стили

HEADER\_STYLE = "BoldHeader"

HEADER\_LINK\_STYLE = "BoldHeaderHyperlink"

CONTENT\_STYLE = "Content"

CODE\_STYLE = "Code"

document = Document("TemplateTOC1.docx")

print(CONFIG\_JSON)

config = getConfigFile(CONFIG\_JSON)

tasks = config["tasks"]

i=1

k=-1

for task in tasks:

i=i+1

header = f"{task['topic']}"

Text = f"{task['Text']}"

print("i: ", i)

print("header: ", header)

print("Text: ", Text)

section = document.AddSection()

paragraph = section.AddParagraph()

paragraph = section.AddParagraph()

paragraph.AppendText(header)

paragraph.ApplyStyle(BuiltinStyle.Heading2)

if header == "Содержание":

paragraph.Format.HorizontalAlignment = HorizontalAlignment.Center

print("header Normal: ", header)

if Text != " " and header != "Моделирование":

#if Text != " " and header != "Моделирование":

models = task["models"]

print("header Heading2: ", header)

print("models: ", models)

if models == " ":

paragraph = section.AddParagraph()

paragraph.AppendText("{{" + Text + "}}")

paragraph.Format.HorizontalAlignment = HorizontalAlignment.Left

else:

paragraph = section.AddParagraph()

paragraph.AppendText(models)

paragraph.Format.HorizontalAlignment = HorizontalAlignment.Left

paragraph = section.AddParagraph()

if header == "Моделирование":

models = task["models"]

for m in models:

k=k+1

Rname = f"Model: {m}"

paragraph = section.AddParagraph()

paragraph.AppendText(Rname)

paragraph.ApplyStyle(BuiltinStyle.Heading3)

paragraph = section.AddParagraph()

paragraph.AppendText("{{ClassBlk" + str(k) + "}}")

paragraph.Format.HorizontalAlignment = HorizontalAlignment.Left

paragraph = section.AddParagraph()

paragraph.Format.HorizontalAlignment = HorizontalAlignment.Center

text = paragraph.AppendText("Талица классификации")

text.CharacterFormat.FontName = "Times New Roman"

text.CharacterFormat.FontSize = 12

text.CharacterFormat.Bold = True

paragraph = section.AddParagraph()

paragraph.Format.HorizontalAlignment = HorizontalAlignment.Center

AddBookmark("t" + str(k), "Table" + str(k))

paragraph = section.AddParagraph()

paragraph = section.AddParagraph()

paragraph.Format.HorizontalAlignment = HorizontalAlignment.Center

text = paragraph.AppendText("Confusion Matrix")

text.CharacterFormat.FontName = "Times New Roman"

text.CharacterFormat.FontSize = 12

text.CharacterFormat.Bold = True

paragraph = section.AddParagraph()

paragraph.Format.HorizontalAlignment = HorizontalAlignment.Center

paragraph.AppendText("{{Heatmap" + str(k) + "}}")

paragraph = section.AddParagraph()

paragraph = section.AddParagraph()

paragraph.Format.HorizontalAlignment = HorizontalAlignment.Center

text = paragraph.AppendText("ROC Curve")

text.CharacterFormat.FontName = "Times New Roman"

text.CharacterFormat.FontSize = 12

text.CharacterFormat.Bold = True

paragraph = section.AddParagraph()

paragraph.Format.HorizontalAlignment = HorizontalAlignment.Center

paragraph.AppendText("{{PltR" + str(k) + "}}")

paragraph = section.AddParagraph()

paragraph = section.AddParagraph()

paragraph.Format.HorizontalAlignment = HorizontalAlignment.Center

text = paragraph.AppendText("Score plot")

text.CharacterFormat.FontName = "Times New Roman"

text.CharacterFormat.FontSize = 12

text.CharacterFormat.Bold = True

paragraph = section.AddParagraph()

paragraph.Format.HorizontalAlignment = HorizontalAlignment.Center

paragraph.AppendText("{{PltL" + str(k) + "}}")

outputFile = "TemplateTOC.docx"

document.SaveToFile(outputFile, FileFormat.Docx)

document.Close()

# In[6]:

path = Path(r"C:\IPYNBgesamt\ASNI-FEN\ASNI-Report")

update\_toc(str(path) + "\TemplateTOC.docx")

# In[7]:

from docx import Document, enum

doc = Document("TemplateTOC.docx")

lines = doc.paragraphs

for line in lines:

#print(line)

if "{{ClassBlk" in line.text or "{{Kap" in line.text:

print(line.text)

line.paragraph\_format.first\_line\_indent = Inches(0.25)

continue

doc.save("rReportTest.docx")

# In[8]:

doc = DocxTemplate("rReportTest.docx")

reportWordPath = 'ASNI\_ReportTest01.docx'

Nk=18

with open("Templates/ASNIR.json", "w", encoding="utf-8") as file\_handle:

json.dump(Asni, file\_handle, indent=4)

with open('Templates/ASNIR.json', 'r', encoding='utf-8') as file\_object:

ASNI\_dict = json.load(file\_object)

ASNI\_dict['logo'] = InlineImage(doc, 'C:\IPYNBgesamt\ASNI-FEN\ASNI-Report\ASSETS\GRAPH0.png', Cm(12))

for num in range(11):

i=num

#print(i)

with open("data/rKap0" + str(i) + ".txt", encoding='UTF-8') as f:

rTxt = f.read()

ASNI\_dict['Kap0' + str(i)] = rTxt

for num in range(Nk):

i=num+1

with open("data/rMod" + str(i) + ".txt") as f:

Txt = f.read()

ASNI\_dict['ClassBlk'+ str(num)] = Txt

ASNI\_dict['Heatmap' + str(num)] = InlineImage(doc, 'C:\IPYNBgesamt\ASNI-FEN\ASNI-Report\ASSETS\Heatmap' + str(num) + '.png', Cm(18))

ASNI\_dict['PltR' + str(num)] = InlineImage(doc, 'C:\IPYNBgesamt\ASNI-FEN\ASNI-Report\ASSETS\PlotROC' + str(num) + '.png', Cm(18))

ASNI\_dict['PltL' + str(num)] = InlineImage(doc, 'C:\IPYNBgesamt\ASNI-FEN\ASNI-Report\ASSETS\plot\_learning\_curve' + str(num+1) + '.png', Cm(20))

#print(ASNI\_dict)

doc.render(ASNI\_dict)

doc.save(reportWordPath)

reportWordPath="ASNI\_ReportTest01.docx"

doc.save("ASNI\_ReportTest01.docx")

# In[9]:

word = win32com.client.DispatchEx("Word.Application")

word.Visible = 1

doc = word.Documents.Open(str(path) + "\ASNI\_ReportTest01.docx")

i=1

for num in range(Nk):

i=i+1

df=pd.read\_csv(f'data/CLSB\_{num}.csv')

rng = doc.Bookmarks("Table" + str(num)).Range

Table=rng.Tables.Add(rng,NumRows=df.shape[0]+1,NumColumns=df.shape[1])

for col in range(df.shape[1]):

Table.Cell(1,col+1).Range.Text=str(df.columns[col])

for row in range(df.shape[0]):

Table.Cell(row+1+1,col+1).Range.Text=str(df.iloc[row,col])

doc.Close()

word.Quit()

print("Table in Ordnung!")

# In[10]:

document = Document()

word\_path="C:\IPYNBgesamt\ASNI-FEN\ASNI-Report\ASNI\_Report.docx"

doc = Document("ASNI\_ReportTest01.docx")

for table in doc.tables:

table.alignment = WD\_TABLE\_ALIGNMENT.CENTER

table.autofit = False

table.allow\_autofit = False

n\_rows=len(table.rows)

n\_cols=len(table.columns)

table.cell(0, 0).text = 'Classes+Metrics'

table.add\_row()

set\_column\_width(table, 0, 35)

for c in range(1, 5):

set\_column\_width(table, c, 20)

g = table.cell(n\_rows, 0)

h = table.cell(n\_rows, n\_cols-1)

g.merge(h)

cell = table.cell(n\_rows, n\_cols-1)

cell.paragraphs[0].paragraph\_format.space\_before = Inches(0)

cell.paragraphs[0].alignment = WD\_PARAGRAPH\_ALIGNMENT.LEFT

cell.paragraphs[0].add\_run("© Dr. Alexander Wagner. Все права охраняются законом")

change\_table\_cell(table.rows[n\_rows].cells[2], background\_color="lightgreen", font\_color="0000ff", font\_size=8, bold=True, italic=True)

table.style = 'Table Grid'

for i in range(1, n\_rows):

for j in range(1, n\_cols):

element=table.cell(i, j).text

partition = element.partition('.')

if (partition[0].isdigit() and partition[1] == '.' and partition[2].isdigit()):

newelement = float(element)

y=round(newelement,3)

table.cell(i, j).text=str(y)

table.cell(i, j).paragraphs[0].paragraph\_format.alignment = WD\_TABLE\_ALIGNMENT.RIGHT

for c in range(0, n\_cols):

change\_table\_cell(table.rows[0].cells[c], background\_color="lightgreen", font\_color="0000ff", font\_size=12, bold=True, italic=True)

for cell in table.columns[c].cells:

cell.paragraphs[0].paragraph\_format.space\_after = Inches(0)

cell.paragraphs[0].paragraph\_format.space\_before = Inches(0)

cell.vertical\_alignment = WD\_CELL\_VERTICAL\_ALIGNMENT.CENTER

set\_cell\_border(

cell,

top={"sz": 0.5, "val": "double", "color": "#000000", "space": "0"},

bottom={"sz": 0.5, "val": "double", "color": "#000000", "space": "0"},

left={"sz": 0.5, "val": "double", "color": "#000000", "space": "0"},

right={"sz": 0.5, "val": "double", "color": "#000000", "space": "0"},

insideH={"sz": 0.5, "val": "double", "color": "#000000", "space": "0"},

end={"sz": 0.5, "val": "double", "color": "#000000", "space": "0"}

)

for row in table.rows:

row.height = Cm(0.55)

row.height\_rule = WD\_ROW\_HEIGHT\_RULE.EXACTLY

table.rows[0].height = Cm(0.6)

table.rows[0].height\_rule = WD\_ROW\_HEIGHT\_RULE.EXACTLY

table.rows[n\_rows-1].height = Cm(0.45)

table.rows[n\_rows-1].height\_rule = WD\_ROW\_HEIGHT\_RULE.EXACTLY

doc.save("ASNI\_Report.docx")

print("ASNI\_Report.docx fertig!")

# ### Programm-Block Приложение (в разработке)

# In[11]:

from spire.doc import \*

document = Document()

file = os.path.join(cwd, "ASNI\_Report.docx")

dt=datetime.datetime.fromtimestamp(os.stat(file).st\_mtime)

txd = "Документ актуализирован: " + dt.strftime('%d.%m.%Y %H:%M:%S')

print("Mody:", txd)

# Load a Word document

document.LoadFromFile(file)

# Get the first section

section = document.Sections[0]

# Get header

header = section.HeadersFooters.Header

# Add a paragraph to the header and set its alignment style

headerParagraph = header.AddParagraph()

headerParagraph.Format.HorizontalAlignment = HorizontalAlignment.Left

#headerParagraph.Format.VerticalAlignment = VerticalAlignment.Center

section.header\_distance = Cm(1.2)

headerPicture = headerParagraph.AppendPicture("ASSETS\logo2.jpg")

headerPicture.TextWrappingStyle = TextWrappingStyle.Square

headerPicture.VerticalOrigin = VerticalOrigin.Line

headerPicture.VerticalAlignment = ShapeVerticalAlignment.Center

#headerPicture.HorizontalAlignment = ShapeHorizontalAlignment.Right

headerPicture.HorizontalAlignment = ShapeHorizontalAlignment.Left

headerPicture.VerticalOrigin = VerticalOrigin.TopMarginArea

text = headerParagraph.AppendText("Автоматизировання Система Научных Исследований в медицине и здравоохранении «АСНИ-МЕД»")

text.CharacterFormat.FontName = "Times New"

text.CharacterFormat.FontSize = 9

text.CharacterFormat.Bold = True

text.CharacterFormat.TextColor = Color.get\_Blue()

section = document.Sections[0]

# Get footer

footer = section.HeadersFooters.Footer

# Add a paragraph to the footer paragraph and set its alignment style

footerParagraph = footer.AddParagraph()

footerParagraph.Format.HorizontalAlignment = HorizontalAlignment.Left

# Add text to the footer paragraph and set its font style

text = footerParagraph.AppendText("© Dr. Alexander Wagner, Все права охраняются законом. " + txd)

text.CharacterFormat.FontName = "Times New"

text.CharacterFormat.FontSize = 9

text.CharacterFormat.Bold = True

text.CharacterFormat.TextColor = Color.get\_Blue()

footerParagraph = footer.AddParagraph()

footerParagraph.Format.HorizontalAlignment = HorizontalAlignment.Right

text = footerParagraph.AppendText("Page ")

txt1=footerParagraph.AppendField("page number", FieldType.FieldPage)

txt2=footerParagraph.AppendText(" of ")

txt3=footerParagraph.AppendField("number of pages", FieldType.FieldNumPages)

text.CharacterFormat.TextColor = Color.get\_Blue()

txt1.CharacterFormat.TextColor = Color.get\_Blue()

txt2.CharacterFormat.TextColor = Color.get\_Blue()

txt3.CharacterFormat.TextColor = Color.get\_Blue()

# Save the result file

document.SaveToFile("AddFootnoteForParagraph.docx", FileFormat.Docx2016)

document.Close()

# In[12]:

from docx import Document

#import time

doc = Document("AddFootnoteForParagraph.docx")

s=len(doc.sections)

for nt in range(s):

section = doc.sections[nt]

header = doc.sections[nt].header

footer = doc.sections[nt].footer

section.header\_distance = Cm(1.0)

section.footer\_distance = Cm(1.0)

header\_para = header.paragraphs[0]

header\_para.paragraph\_format.space\_before = Pt(0)

header\_para.paragraph\_format.space\_after = Pt(7)

footer\_para = footer.paragraphs[0]

footer\_para.paragraph\_format.space\_before = Pt(0)

footer\_para.paragraph\_format.space\_after = Pt(0)

footer\_para = footer.paragraphs[1]

footer\_para.paragraph\_format.space\_before = Pt(0)

footer\_para.paragraph\_format.space\_after = Pt(0)

section = doc.sections[0]

section.different\_first\_page\_header\_footer = True

lines = doc.paragraphs

n=-1

for line in lines:

n=n+1

if line.text == "Evaluation Warning: The document was created with Spire.Doc for Python.":

delete\_paragraph(line)

continue

reportWordPath="ASNI\_ReportPre.docx"

doc.save("ASNI\_ReportPre.docx")

print("ASNI\_ReportPre.docx fertig!")

# In[13]:

Text\_copy(r"C:\IPYNBgesamt\ASNI-FEN\ASNI-Report\data\Vorwort.docx")

time.sleep(2.4)

replace\_copy("C:\IPYNBgesamt\ASNI-FEN\ASNI-Report\ASNI\_ReportPre.docx", "xHier")

time.sleep(4.4)

Text\_copy(r"C:\IPYNBgesamt\ASNI-FEN\ASNI-Report\data\biblioTestKrollAu.docx")

time.sleep(4.4)

replace\_copy("C:\IPYNBgesamt\ASNI-FEN\ASNI-Report\ASNI\_ReportResult.docx", "xLiter")

print("Programm Insert-Blöcke beendet!")

print("ASNI\_ReportResult.docx fertig!")

# In[14]:

path = Path(r"C:\IPYNBgesamt\ASNI-FEN\ASNI-Report")

update\_toc(str(path) + "\ASNI\_ReportResult.docx")

# In[15]:

from docx import Document

reportWordPath = os.path.join(cwd, "ASNI\_ReportResult.docx")

reportOutPath = os.path.join(cwd, "ASNI\_ReportV05R4.docx")

print("reportWordPath: ",reportWordPath)

print("reportOutPath: ", reportOutPath)

doc = Document(reportWordPath)

doc.save(reportOutPath)

print("Printed immediately2.4")

time.sleep(2.4)

print("Printed after 2.4 seconds.")

convert(reportOutPath, reportOutPath.replace(".docx", ".pdf"))

print(reportOutPath + " fertig!")

now = datetime.datetime.now()

timeend = now.replace(microsecond=0)

print("Programm Ende: ", timeend)

timedelta = (timeend-timestart)

print ("Programm Teil II dauert: " + str(timedelta) + " seconds")

----------------- End of File! -----------------