ASSIGNMENT 1 MATH 944:DATA SCIENCE FOR STATISTICIANS

CHUKA UNIVERSITY

SD18/79234/25: JAMES MWITI MUTEGI

QUESTION 1

Data Science Workflow for Public Policy

The data science workflow for public policy entails interconnected stages, each requiring careful attention to transparency, reproducibility, and ethical considerations.

Here's a comprehensive breakdown

Problem Definition & Stakeholder Engagement

Key Activities: Meet with county officials, understand policy objectives, define success metrics, and identify constraints.

Trustworthiness Action: Create a formal project charter document that explicitly states the following

- The policy question being addressed
- Success criteria and performance metrics
- Identified stakeholders and their concerns
- Ethical considerations and potential biases
- Data limitations and scope boundaries

Data Collection & Acquisition

Key Activities: Identify data sources, assess data availability, gather datasets from various sources (surveys, administrative records, sensors, etc.)

Trustworthiness Action: Maintain a comprehensive data lineage document that records the following information

- Source of each dataset (URL, contact person, date accessed)
- Data collection methodology
- Known limitations or biases in data collection
- Legal and ethical clearances for data use
- Any sampling procedures used

This allows county officials to verify data provenance and understand what the data represents (and doesn't represent).

Data Cleaning & Preprocessing

Key Activities: Handle missing values, remove duplicates, standardize formats, address outliers, create derived variables.

Trustworthiness Action: Write version-controlled, well-commented code with explicit data transformation logs that ensures that the following is met

- Documents every data modification (what changed, why, and when)
- Uses programmatic approaches rather than manual Excel edits
- Includes data validation checks at each step
- Exports a transformation report showing before/after statistics This creates a complete audit trail allowing officials to see exactly how raw data became analysis-ready data.

Exploratory Data Analysis (EDA)

Key Activities: Generate summary statistics, create visualizations, identify patterns, test initial hypotheses, check assumptions.

Trustworthiness Action: Conduct and document a systematic bias and fairness audit that:

- Disaggregates analysis by demographic groups (race, income, geography)
- Checks for representation gaps in the data
- Identifies potential disparate impacts
- Examines historical inequities that might be encoded in the data

This ensures the analysis doesn't inadvertently perpetuate existing inequalities or overlook vulnerable populations.

Feature Engineering & Selection

Key Activities: Create meaningful variables, transform features, select relevant predictors, reduce dimensionality.

Trustworthiness Action: Document the theoretical justification for each feature by creating a feature codebook that includes the following

a. Clear operational definitions b. Policy relevance (why this matters for decision-making) c. Known correlations and potential confounders d. Domain expert validation of feature appropriateness

This prevents "black box" feature selection and ensures variables have substantive policy meaning, not just statistical correlation.

Model Development & Training

Key Activities: Select appropriate analytical methods, train models, tune hyperparameters, validate assumptions.

Trustworthiness Action: Use cross-validation with temporal or geographic holdout sets that:

a. Tests whether the model generalizes to unseen data b. Validates performance across different county regions or time periods c. Documents model performance metrics with confidence intervals d. Compares multiple modeling approaches transparently

This demonstrates that findings aren't artifacts of overfitting and that the model works in realistic conditions.

Model Evaluation & Interpretation

Key Activities: Assess model performance, conduct sensitivity analyses, interpret coefficients/feature importance, test robustness.

Trustworthiness Action: Perform comprehensive sensitivity and robustness analyses that:

a. Tests how results change with different modeling assumptions b. Examines impact of removing influential observations c. Varies key parameters to establish confidence bounds d. Documents scenarios where recommendations might not hold

This provides honest uncertainty quantification and helps officials understand when the analysis is most/least reliable.

Communication & Visualization

Key Activities: Create policy briefs, develop visualizations, prepare presentations, translate technical findings.

Trustworthiness Action: Develop a multi-level documentation strategy including:

Executive summary for decision-makers (non-technical) Technical appendix with full methodology Interactive dashboard for exploring results Plain-language limitations section explaining what the analysis cannot tell you

This ensures transparency while making findings accessible to audiences with varying technical expertise.

Deployment & Monitoring

Key Activities: Implement recommendations, create decision-support tools, establish monitoring systems.

Trustworthiness Action: Establish a structured monitoring and feedback protocol that:

Defines key performance indicators to track post-implementation Creates a schedule for model retraining/updating Establishes thresholds that trigger model review Includes mechanism for stakeholder feedback on real-world performance

This ensures the analysis remains valid over time and that unintended consequences are detected early.

Reproducibility & Knowledge Transfer

Key Activities: Archive code and data, document processes, train county staff, enable future replication.

Trustworthiness Action: Create a complete reproducibility package that includes:

Version-controlled code repository (GitHub/GitLab) README with step-by-step execution instructions Containerized computing environment (Docker) or requirements.txt Sample data for testing the pipeline License information and data use agreements

This allows any qualified analyst to reproduce the entire analysis from scratch, which is essential for public accountability.

QUESTION 2

Comparison of R and Python

Data Cleaning

R (Tidyverse – dplyr, tidyr):

Pros:

Intuitive syntax for data manipulation (filter, mutate, select).

Cons:

May have steeper learning curve for those not familiar with R.

Python (pandas):

Pros:

- Extremely popular and widely used for data cleaning (functions like .groupby(), .fillna(), .dropna(), .apply()).
- Easily integrates with both scripting and software engineering workflows.
- Supported by excellent documentation and community resources. Cons: - -- Slightly verbose syntax compared to some tidyverse shortcuts, but highly flexible.

Visualization (for Non-Technical Audience)

R (ggplot2):

Pros:

- Well known for beautiful, publication-quality graphics.

Cons:

More of a data analyst/statistician audience; less direct tools for interactive web deployment.

Python (matplotlib, seaborn, plotly):

Pros:

seaborn:

High-level statistical graphics, great for quickly plotting data with sensible defaults.

matplotlib:

Full control for customization.

plotly:

Enables creation of interactive, web-friendly visualizations suitable for sharing with non-technical audiences (drag-and-drop, tooltips, dashboards).

Cons:

May require combining packages for best results, but integration is simple.

Regression Modeling

R (stats, caret):

Pros:

Strong support for statistical models; intuitive for those familiar with statistical terminology.

Cons:

Can be challenging for more complex machine learning workflows

Python (scikit-learn, statsmodels):

Pros:

scikit-learn:

Versatile library for building, training, and evaluating regression models. Supports linear regression, decision trees, and more.

statsmodels

For statistical regression models and robust diagnostics. Easy to standardize workflows, automate feature engineering, and integrate into larger analytic pipelines. Cons: Requires understanding of NumPy concepts, though well-documented.

Why Choose Python Over R?

1. Integration Across Workflow:

Python can be used for scripting, building dashboards, and developing entire web apps. This makes it ideal if you ever need to automate your analysis, scale up, or integrate with existing IT infrastructure.

2. Packages for Every Step:

pandas for data cleaning and manipulation seaborn, matplotlib, plotly for clear, publication-quality, and interactive visualizations scikit-learn and statsmodels for building and evaluating regression models

3. Broad Community & Ecosystem:

Python's vast community ensures constant support, frequent updates, and compatibility with modern tools (e.g., Jupyter notebooks, web deployment via Flask/Dash).

4. Non-Technical Audience Engagement:

Python's interactive visualization tools (plotly, Dash) allow creation of dashboards and interactive reports, essential for communicating with non-technical stakeholders.

5. General-Purpose Nature:

Python is a general-purpose programming language, enabling you to move beyond analysis—into automation, API development, and more.

QUESTION 3

CREATION OF PROJECT DIRECTORY

In [ ]:
```
mkdir nairobi-clinic-wait-times
cd nairobi-clinic-wait-times
```

2. **Folder Structure:**
```
nairobi-clinic-wait-times/
│
├── data/
│   ├── raw/
│   ├── processed/
│   └── README.md
│
├── notebooks/
│   ├── 01_data_cleaning.ipynb
│   ├── 02_exploratory_analysis.ipynb
│   ├── 03_regression_model.ipynb
│   └── 04_visualization.ipynb
│
├── src/                    # Python modules
│   ├── __init__.py
│   ├── data_processing.py
│   └── utils.py
│
├── outputs/
│   ├── figures/
│   ├── tables/
│   └── reports/
│
├── tests/                  # Unit tests
│   └── test_data_processing.py
│
├── requirements.txt        # Package dependencies
├── environment.yml         # Conda environment (alternative)
├── .gitignore
└── README.md
```

CREATING FOLDER

In [3]:
```
# mkdir -p data/{raw,processed} notebooks src outputs/{figures,tables,reports} tests
```

Importing data

In [6]:
```
# install the libraries if not installed by the command
#!pip install (library)
import pandas as pd
import numpy as np
from datetime import datetime
import matplotlib.pyplot as plt
import seaborn as sns
```

Loading data

In [24]:
```
 #file_path = r"C:\Users\User\OneDrive\Desktop\r\Mdata.csv"
```

In [26]:
```
#data=pd.read_csv(file_path)
#data
```

Data cleaning

In [13]:
```
#############Create a copy
#clinic_clean = clinic_data.copy()

###############Clean column names
#clinic_clean.columns = clinic_clean.columns.str.lower().str.replace(' ', '_')

################Remove duplicates
#clinic_clean = clinic_clean.drop_duplicates()

################## Handle missing values
#clinic_clean = clinic_clean.dropna(subset=['wait_time_minutes', 'service_outcome'])

##################### Convert dates
#clinic_clean['visit_date'] = pd.to_datetime(clinic_clean['visit_date'])

######################### Validate ranges
#clinic_clean = clinic_clean[
   # (clinic_clean['wait_time_minutes'] >= 0) &
   # (clinic_clean['wait_time_minutes'] <= 480)
#]

############################## Save cleaned data
#clinic_clean.to_csv('../data/processed/clinic_data_clean.csv', index=False)

#print(f"Cleaned dataset shape: {clinic_clean.shape}")
```

For another researcher

In [14]:
```
# create environment file
#conda env create -f environment.yml
#conda activate nairobi-clinic-analysis
```

In [15]:
```
import sys
print(f"Python version: {sys.version}")
```

```
Python version: 3.13.5 | packaged by Anaconda, Inc. | (main, Jun 12 2025, 16:37:03) [MSC v.1929 64 bit (AMD64)]
```

In [17]:
```
# Last cell of notebook
#import pkg_resources
#packages = ['pandas', 'numpy', 'scikit-learn']
#for package in packages:
    #print(f"{package}: {pkg_resources.get_distribution(package).version}")
```

QUESTION 4

Version Control for Accountability

How Git Contributes to Reproducibility Initializing a Git repository provides:

A. Complete Change History:

Every modification to analysis scripts is tracked with timestamp and author Can review what changed, when, and why through commit messages Enables auditing of analytical decisions

B. Accountability:

Clear attribution of who made which changes County officials can trace analytical decisions back to specific team members Creates professional documentation trail for public sector work

C. Rollback Capability:

If an error is introduced, can revert to any previous working state Prevents loss of work Allows safe experimentation

D. Collaboration:

Multiple team members can work simultaneously Changes can be merged systematically Reduces risk of overwriting colleagues' work

E. Transparency:

Remote repository (GitHub) provides public or controlled access County officials can review analysis development process Supports open science principles for public policy research

F. Reproducibility:

Anyone can clone repository and recreate exact analysis Tagged versions (e.g., "v1.0-final-report") mark specific states Branch system allows testing alternative approaches without affecting main analysis

In [19]:
```
################ First, configure Git with your identity
# Set your name and email (visible in commit history)
########git config --global user.name "Your Name"
#git config --global user.email "mwitiwamutegi.email@nairobi.go.ke"

###########Optional: Set default branch name to 'main'
#git config --global init.defaultBranch main
```

Initialize Repository

In [21]:
```
########### Navigate to project directory
#cd nairobi-clinic-wait-times

############### Initialize Git repository
#git init

###### This creates a hidden .git folder that stores version history
##```

##Expected output:
##```
##Initialized empty Git repository in /path/to/nairobi-clinic-wait-times/.git/
```

Create .gitignore File

Before tracking files, create .gitignore to exclude sensitive or unnecessary files:

In [23]:
```
# Track a newly created script called data_cleaning.py
# Check current status (see untracked files)
###
#git status
#```

#**Expected output:**
#```
#On branch main

#No commits yet

#Untracked files:
 # (use "git add <file>..." to include in what will be committed)
#       scripts/data_cleaning.py
#       requirements.txt
#       .gitignore

# nothing added to commit but untracked files present (use "git add" to track)
#git status shows which files are untracked (red), modified, or staged (green)
#git add moves the file to the "staging area" (preparing it for commit)
#The file is now tracked but not yet permanently saved in the repository
```

Save (commit) the current state of the script with a descriptive message

In [28]:
```
# Commit staged changes with a descriptive message
#git commit -m "Add initial data cleaning script for Nairobi clinic wait time analysis"
#```

#**Expected output:**
#```
#[main (root-commit) 3a7f891] Add initial data cleaning script for Nairobi clinic wait time analysis
 #1 file changed, 67 insertions(+)
# create mode 100644 scripts/data_cleaning.py
#git commit permanently saves the staged changes to the repository
#-m "message" provides a commit message inline
#The commit creates a unique identifier (hash like 3a7f891)
# Shows how many files changed and lines added/removed
```

Upload (push) committed changes to a remote repository on GitHub

Step 3a: Create Remote Repository on GitHub Go to GitHub: Navigate to https://github.com Create new repository

Click the "+" icon (top-right) → "New repository" Repository name: nairobi-clinic-wait-times Description: "Analysis of patient wait times at Nairobi County public health clinics" Choose: Public (for transparency) or Private (for sensitive analysis) Important: Do NOT initialize with README, .gitignore, or license (since you already have local commits)

In [ ]: