VICTORIA UNIVERSITY
Kampala . Uganda
"Come Prepared to Learn, Leave Prepared to Succeed"

PROGRAMMING WITH DATA SCIENCE (PYTHON) COURSEWORK: DECEMBER 2025

BY BUA ANTHONY

This is an individual assignment. Each student will attempt the question assigned to their course and submit it using V-Class. In addition, students will present their work to the lecturer during in-class sessions. They are required to document all steps taken in completing the assignment, attach relevant codes, papers, diagrams where applicable, and be prepared to defend their work during the presentation.

MSc Big Data Analytics – Python-Based Research Case Study Coursework

Background

The exponential growth of digital data across sectors such as healthcare, finance, smart cities, e-government, and social media has created unprecedented opportunities and challenges for data-driven decision-making. Big Data Analytics provides the tools and methodologies to process, analyze, and extract actionable insights from massive datasets, enabling organizations to make evidence-based decisions, optimize operations, and innovate services.

Python has emerged as the dominant language for Big Data Analytics due to its flexibility, readability, extensive ecosystem of libraries (pandas, NumPy, Matplotlib, Seaborn, scikit-learn, TensorFlow, PySpark, Dask), and seamless integration with modern data processing frameworks. Its capabilities allow data scientists to perform data cleaning, transformation, visualization, statistical analysis, machine learning, and scalable computation on large and complex datasets.

Replicating existing research using Python is a critical practice in the field of data science. It ensures reproducibility, a fundamental principle in scientific research, allowing researchers to validate methodologies, verify results, and identify potential gaps. Replication also provides an opportunity to enhance or extend existing models, either by improving predictive

accuracy, increasing scalability for large datasets, or incorporating novel analytical techniques.

In many real-world scenarios, datasets are heterogeneous, high-dimensional, and prone to challenges such as missing values, class imbalance, noise, and unstructured formats. Traditional or naïve data analysis approaches are often insufficient to handle such complexities effectively. Leveraging advanced Python-based tools and libraries enables data scientists to design robust workflows, perform comprehensive exploratory data analysis (EDA), implement sophisticated machine learning algorithms, and visualize results meaningfully.

This coursework simulates a real-world research workflow in Big Data Analytics. You are tasked with replicating a contemporary Python-based study, critically analyzing its methodology and findings, and proposing enhancements to improve insights, predictive performance, or scalability. Through this exercise, you will strengthen your technical skills in Python, analytical thinking, and research evaluation.

Task 1: Understanding Existing Research

Your journey begins with understanding how others have applied Python in Big Data Analytics research.

Select one recent scholarly research paper (published between 2020 and 2025) that applies Python in Big Data Analytics in domains such as healthcare, finance, smart cities, or social data. Refer to google scholar, or papers with code or GitHub websites

a) Provide a summary of the paper's key objectives, dataset(s), Python-based methodology (data preprocessing, modeling, visualization), and findings.

b) Replicate the python code, workflow, data pipeline, or diagrams presented in the paper.

c) Critically analyze the paper to identify research gaps or limitations. Based on the identified gaps, propose a solution or enhancement describing how your idea could strengthen the study's methodology, predictive performance, or analytical insights.

Task 2: Define Your Research Context

Now that you have learned from existing work, define your own research scenario.

a) Describe the dataset you are analyzing, including its source, size, types of features (numerical, categorical, textual), and relevance to the chosen domain.

Task 3: Develop and Implement Your Algorithm or Model

Building on the dataset prepared and explored in Task 2, **you are required to design, implement, and evaluate your own algorithm or model using Python**. This task should follow a research methodology style, including clear description, rationale, implementation, and evaluation.

a) Problem Definition and Research Objective

- Clearly define the specific problem your algorithm or model is addressing (e.g., prediction, classification, anomaly detection, clustering, optimization).
- Explain why this problem is significant in your domain and how your solution contributes to knowledge or practical applications.
- State your research objective or hypothesis formally.

b) Proposed Methodology

- Design your algorithm or model in Python. This may involve:
    - Creating a custom function or class to implement a novel approach.
    - Combining existing Python libraries (NumPy, Pandas, scikit-learn, TensorFlow, PyTorch) with your own logic.
    - Using control flow, loops, recursion, or data structures to implement key steps.
- Document your methodology in a research-style format, including pseudocode, flowcharts, or high-level descriptions of the logic before coding. Example:

Algorithm: Custom Anomaly Detection
Input: Dataset X
Output: Anomaly Scores
Step 1: Preprocess dataset (handle missing values, normalize)
Step 2: For each data point:
    Compute distance from cluster mean
    If distance > threshold, flag as anomaly
Step 3: Return flagged anomalies

c) Implementation in Python

- Write clean, well-commented Python code implementing your algorithm.

Example structure:

```
def custom_algorithm(df):
    results = []
    for index, row in df.iterrows():
        score = row['feature1'] * row['feature2']
        if score > threshold:
            results.append(1)  # Flag as positive
        else:
            results.append(0)
    df['result'] = results
    return df
```

- Use data structures such as lists, dictionaries, arrays, or DataFrames to store intermediate computations.
- Apply control flow (if/else, loops) to handle conditions or branching logic in your algorithm.

d) Model Evaluation

- Evaluate your algorithm using appropriate metrics for your problem (e.g., accuracy, F1-score, RMSE, AUC, Silhouette score, precision/recall).
- Visualize results using Python plotting libraries (Matplotlib, Seaborn, or Plotly). Examples: confusion matrix, performance over iterations, or feature importance.
- Compare the performance of your algorithm with a baseline model or approach, highlighting improvements or trade-offs.

e) Research Analysis and Discussion

- Describe how your algorithm addresses the research problem and dataset challenges identified in Task 2.
- Discuss the strengths, limitations, and potential biases of your model.
- Suggest possible enhancements or extensions for future research.

- Ensure your discussion is structured in a research report style, with clear headings for methodology, implementation, evaluation, and interpretation of results.

f) Reproducibility

- Include all code, sample outputs, and visualizations in your Jupyter Notebook or Python script.
- Ensure that another researcher could replicate your results using your dataset and code.

MSc Blockchain – Python-Based Research Case Study Coursework

Background

Block chain technology has transformed digital systems by enabling decentralized, secure, and immutable transaction ledgers across sectors such as finance, healthcare, supply chain, and e-governance. Its adoption has grown rapidly due to its ability to enhance transparency, traceability, and trust in digital processes. However, implementing block chain solutions at scale presents complex challenges, including performance optimization, interoperability between block chain networks, smart contract security, and integration with legacy systems.

Python has emerged as a key language for block chain research and development because of its rich ecosystem for block chain frameworks (e.g., Hyper ledger Fabric, Ethereum via web3.py, Brownie), cryptographic libraries, data analytics, and integration with other technologies such as IoT and Big Data. Python facilitates smart contract testing, block chain simulation, transaction analysis, and performance evaluation.

Replicating existing block chain research studies using Python ensures reproducibility, allows researchers to validate methods, and helps identify improvements in scalability, consensus algorithms, or security mechanisms. Extending these studies enables experimentation with novel solutions such as optimized consensus protocols, hybrid block chain models, or integration of block chain with analytics systems.

This coursework simulates a real-world block chain research workflow, requiring students to replicate a contemporary Python-based block chain study, critically evaluate its methodology, and propose enhancements. You will strengthen your technical skills in block chain, Python programming, and research analysis while learning to develop secure, efficient, and scalable block chain solutions.

Task 1: Understanding Existing Block Chain Research

Your journey begins with understanding how others have implemented block chain in real-world contexts.

Select one recent scholarly research paper (published between 2020 and 2025) that presents a block chain solution using Python in domains such as finance, healthcare, supply chain, or e-governance.

b) Provide a summary of the paper's key objectives, block chain architecture or protocol, Python-based implementation, and findings.

c) Replicate any python code, architecture diagrams, block chain network workflows, or smart contract pipelines presented in the paper.

d) Critically analyze the paper to identify research gaps or limitations. Based on the identified gaps, propose a solution or enhancement describing how your idea could strengthen block chain performance, security, or scalability.

Task 2: Define Your Block Chain Research Context

Now that you have learned from existing work, define your own block chain research scenario.

e) Describe the block chain application you are focusing on, including its domain, network type (public, private, hybrid), consensus mechanism, and integration requirements.

Task 3: Develop and Implement Your Block Chain Algorithm or Model

Building on the dataset, block chain literature, and system context defined in Task 2, you are required to design, implement, and evaluate your own block chain-related algorithm, protocol, or model using Python. This task must follow a research methodology format, demonstrating conceptual clarity, technical implementation, and analytical depth.

Your model may address aspects such as consensus mechanisms, smart contract logic, cryptographic primitives, transaction validation, anomaly detection, block formation, distributed ledger performance, or scalability optimization.

a) Problem Definition and Research Objective

- Clearly define the block chain-specific problem you aim to address. Examples include:
    - Detecting fraudulent block chain transactions.
    - Designing a simplified consensus algorithm.
    - Optimizing block creation time.
    - Modeling peer-to-peer node communication.

    - Simulating network attacks (Sybil, double-spending) and defenses.
- Explain the significance of the problem within block chain systems.
- Present a formal research aim or hypothesis.

b) Proposed Methodology (Block chain + Python)

Design your block chain algorithm or model using Python. You may: - Implement a custom consensus algorithm using Python data structures and control flow. - Design a transaction validation pipeline using lists, dictionaries, hash functions, and conditional statements. - Create a block chain simulation model, including block creation, hashing, and chain validation. - Model a smart contract execution process using classes and functions.

Your methodology must be documented in a formal research format and include pseudocode or a flowchart. Example:

Algorithm: Lightweight Proof-of-Participation (PoP) Consensus
Input: Set of nodes N with participation scores P
Output: Selected leader node L

Step 1: Normalize participation scores
Step 2: Generate random weight distribution
Step 3: For each node:
    Compute selection probability using score * weight
Step 4: Select node with highest probability as leader
Step 5: Append new block signed by leader
Step 6: Broadcast block to network

c) Implementation in Python

Write clean, well-commented Python code implementing your block chain model. Examples:

```
import hashlib
import time

class Block:
  def __init__(self, index, transactions, previous_hash):
    self.index = index
```

```
        self.transactions = transactions
        self.timestamp = time.time()
        self.previous_hash = previous_hash
        self.hash = self.compute_hash()

    def compute_hash(self):
        block_string = f"{self.index}{self.transactions}{self.timestamp}{self.previous_hash}"
        return hashlib.sha256(block_string.encode()).hexdigest()
```

Or a consensus logic example:

```
def simple_pow(difficulty):
    nonce = 0
    while True:
        guess = f"block{nonce}".encode()
        guess_hash = hashlib.sha256(guess).hexdigest()
        if guess_hash.startswith("0" * difficulty):
            return nonce, guess_hash
        nonce += 1
```

Your implementation must: - Use Python data structures (lists, dictionaries, tuples, sets, classes). - Use control flow (if/else, loops, functions, exceptions). - Demonstrate correctness and replicability.

d) Model Evaluation (Block chain Metrics)

Evaluate your block chain model using metrics relevant to your design. Examples: - Security metrics: resistance to tampering, immutability checks, attack simulation. - Performance metrics: block creation time, transaction throughput, latency. - Scalability metrics: node increase vs. time or memory consumption. - Consensus metrics: fairness, leader selection distribution, energy efficiency.

Use Python visualizations to present findings, e.g.: - Block creation time graph. - Node participation distribution. - Hash difficulty vs. time. - Transaction volume vs. validation speed.

e) Research Analysis and Discussion

Provide a deep analysis of your blockchain model: - Explain how the design addresses the problem and challenges previously identified. - Discuss strengths, weaknesses, limitations, and assumptions of your model. - Compare your results to related blockchain research you studied in Task 1. - Propose improvements or future research directions. - Present this analysis in a structured research format.

f) Reproducibility and Research Integrity

- Provide all source code, outputs, and visualizations.
- Ensure your model can be replicated by another researcher.
- Include instructions for running the algorithm or simulation.