Statement and Confirmation of Own Work

A signed copy of this form must be submitted with every assignment.
If the statement is missing your work may not be marked.

Student Declaration

I confirm the following details:

Student Name:	Johnson Kenisha Corera
Student ID Number:	c23020001@cicra.edu.lk
Qualification:	Bachelor in Cyber Security
Unit:	SIT325 Advanced Network Security
Centre:	CICRA Campus
Word Count:	1516

I have read and understood both *Deakin Academic Misconduct Policy* and the *Referencing and Bibliographies* document. To the best of my knowledge my work has been accurately referenced and all sources cited correctly.

I confirm that I have not exceeded the stipulated word limit by more than 10%.

I confirm that this is my own work and that I have not colluded or plagiarized any part of it.

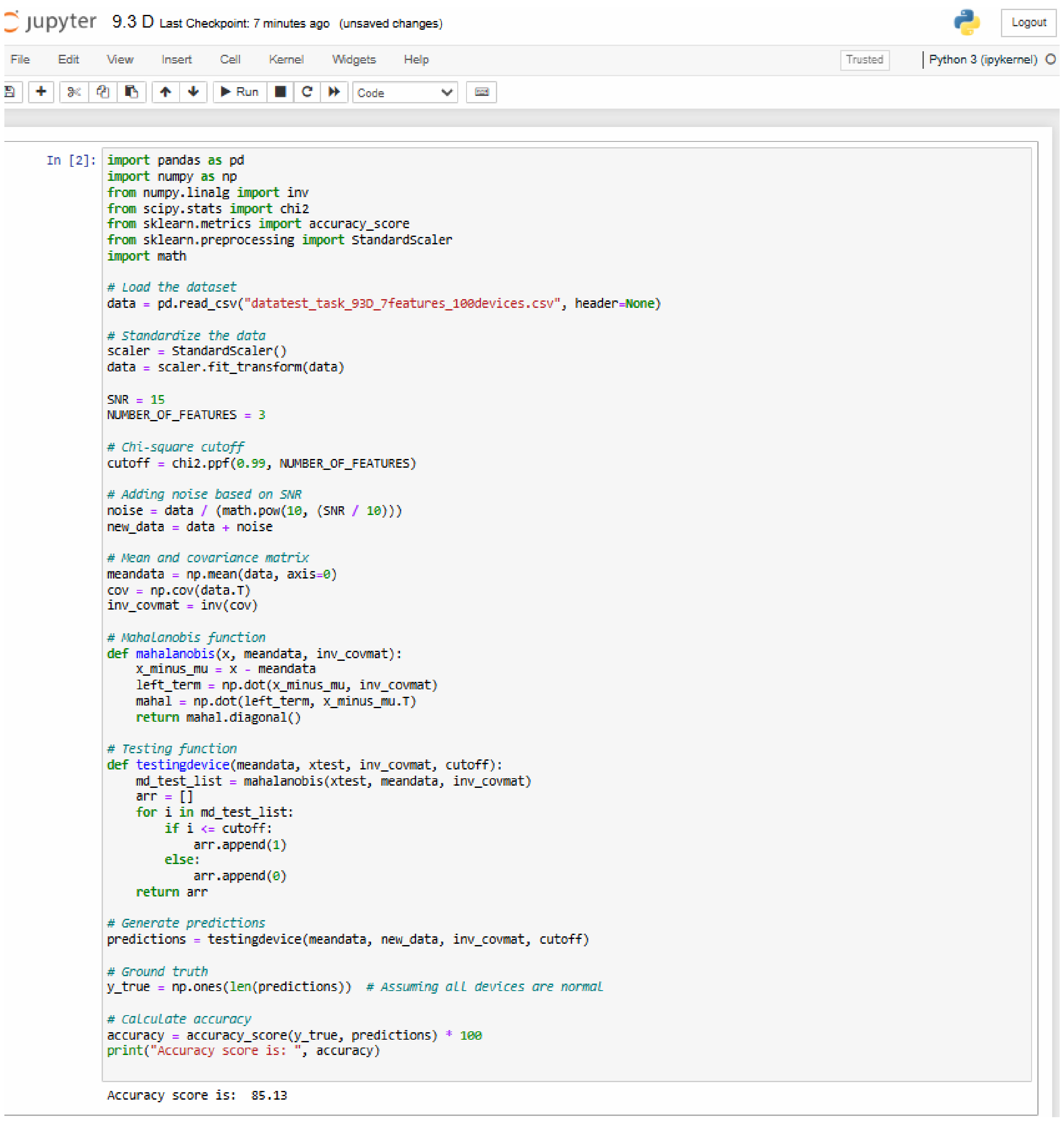
Candidate Signature:	[illegible]
Date:	01/02/2025

Task 9.3D
Table of Content

Table of Figures

Detection Accuracy using 3 Features

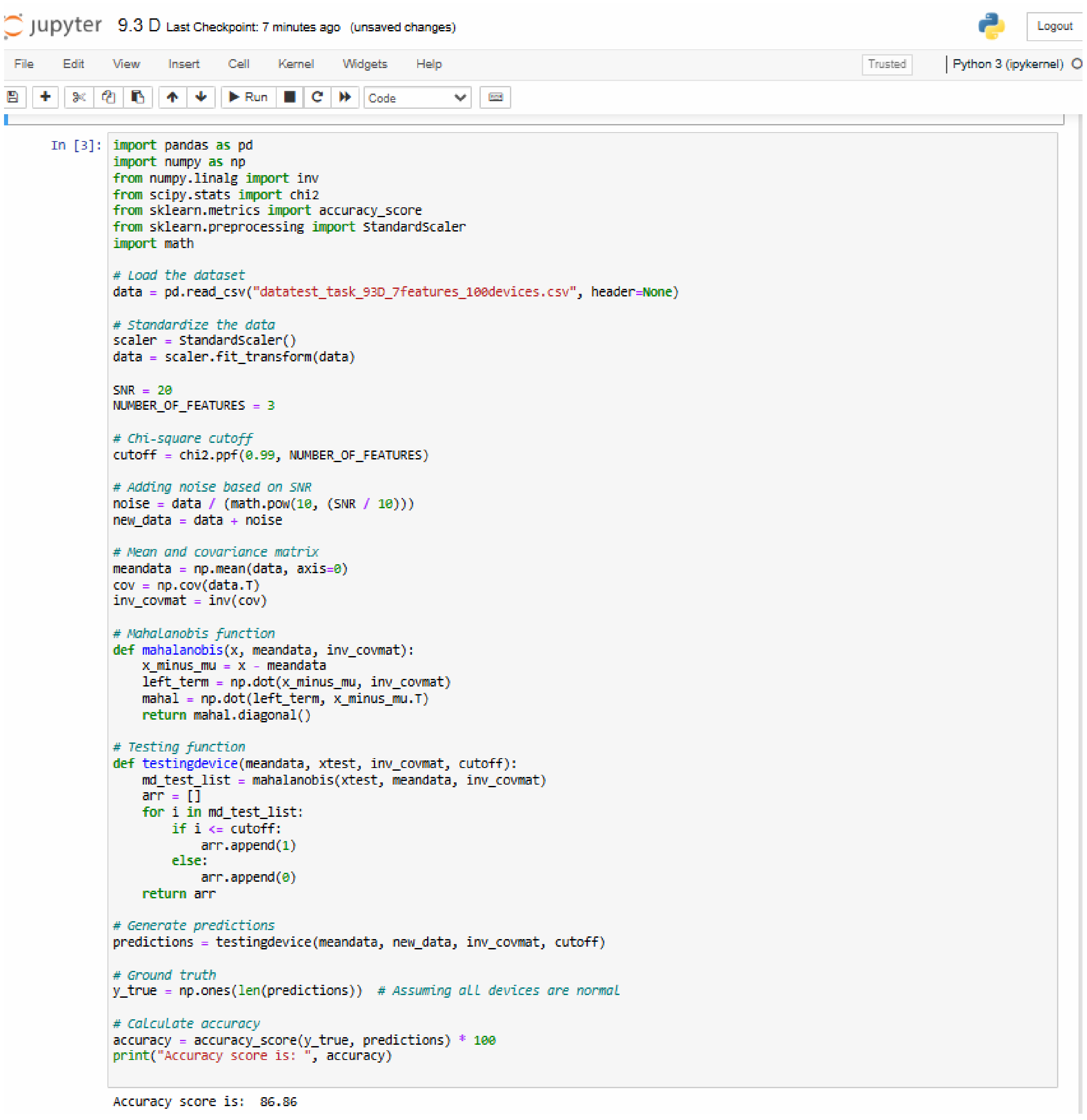
- The following instance showcases detection accuracy when I employ three features and begin the SNR at fifteen before increasing it by five units at each step. The detection accuracy of this system includes three features when operating at SNR level fifteen


jupyter 9.3 D Last Checkpoint: 7 minutes ago (unsaved changes)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3 (ipykernel)
Run
Code

```
In [2]: import pandas as pd
import numpy as np
from numpy.linalg import inv
from scipy.stats import chi2
from sklearn.metrics import accuracy_score
from sklearn.preprocessing import StandardScaler
import math

# Load the dataset
data = pd.read_csv("datatest_task_93D_7features_100devices.csv", header=None)

# Standardize the data
scaler = StandardScaler()
data = scaler.fit_transform(data)

SNR = 15
NUMBER_OF_FEATURES = 3

# Chi-square cutoff
cutoff = chi2.ppf(0.99, NUMBER_OF_FEATURES)

# Adding noise based on SNR
noise = data / (math.pow(10, (SNR / 10)))
new_data = data + noise

# Mean and covariance matrix
meandata = np.mean(data, axis=0)
cov = np.cov(data.T)
inv_covmat = inv(cov)

# Mahalanobis function
def mahalanobis(x, meandata, inv_covmat):
    x_minus_mu = x - meandata
    left_term = np.dot(x_minus_mu, inv_covmat)
    mahal = np.dot(left_term, x_minus_mu.T)
    return mahal.diagonal()

# Testing function
def testingdevice(meandata, xtest, inv_covmat, cutoff):
    md_test_list = mahalanobis(xtest, meandata, inv_covmat)
    arr = []
    for i in md_test_list:
        if i <= cutoff:
            arr.append(1)
        else:
            arr.append(0)
    return arr

# Generate predictions
predictions = testingdevice(meandata, new_data, inv_covmat, cutoff)

# Ground truth
y_true = np.ones(len(predictions))  # Assuming all devices are normal

# Calculate accuracy
accuracy = accuracy_score(y_true, predictions) * 100
print("Accuracy score is: ", accuracy)

Accuracy score is:  85.13
```

Figure 01: SNR 15 and Feature 3

- Detection accuracy using 3 features and a particular SNR of 20

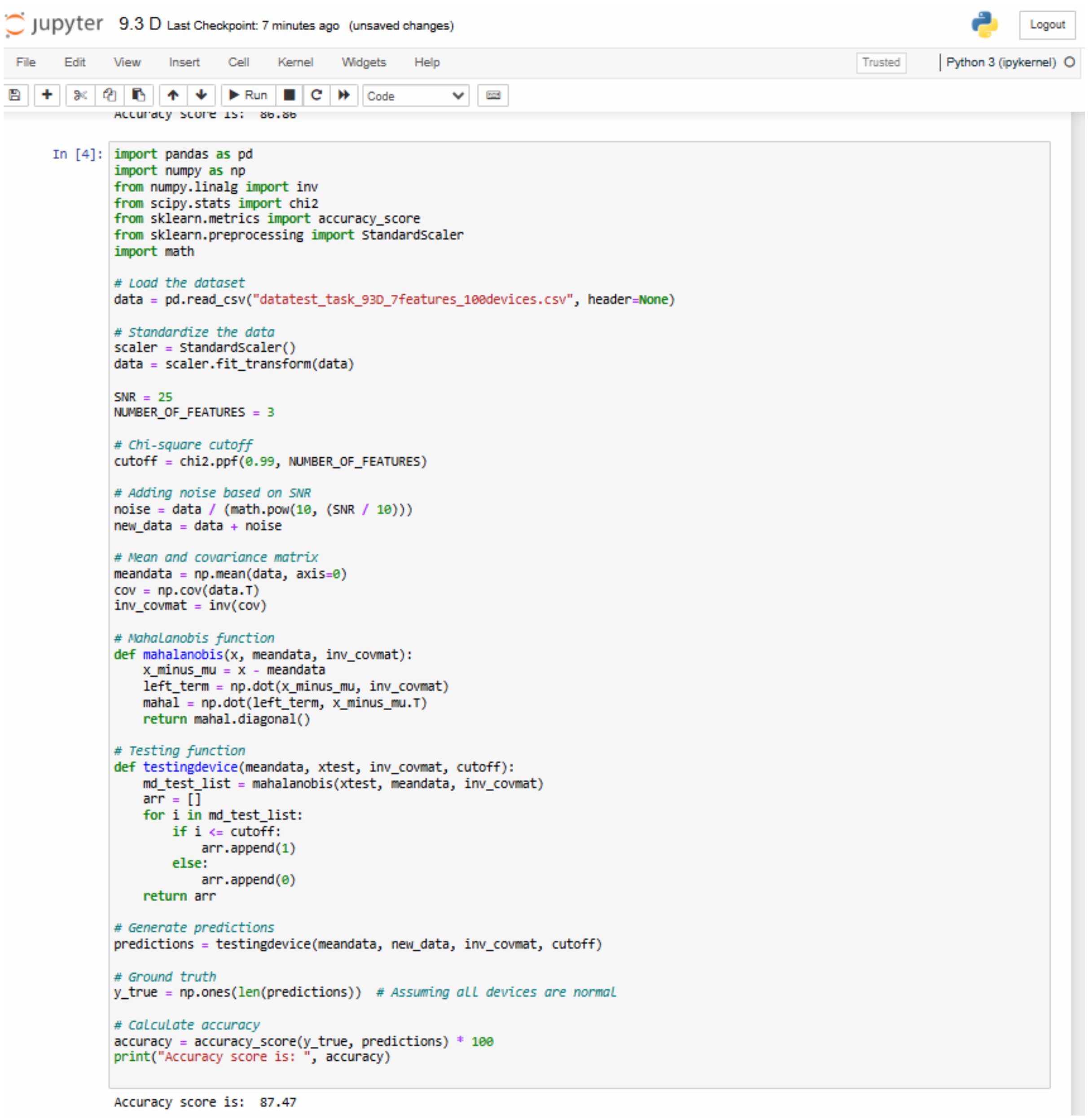

jupyter 9.3 D Last Checkpoint: 7 minutes ago (unsaved changes)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3 (ipykernel)
Run
Code

```
In [3]: import pandas as pd
import numpy as np
from numpy.linalg import inv
from scipy.stats import chi2
from sklearn.metrics import accuracy_score
from sklearn.preprocessing import StandardScaler
import math

# Load the dataset
data = pd.read_csv("datatest_task_93D_7features_100devices.csv", header=None)

# Standardize the data
scaler = StandardScaler()
data = scaler.fit_transform(data)

SNR = 20
NUMBER_OF_FEATURES = 3

# Chi-square cutoff
cutoff = chi2.ppf(0.99, NUMBER_OF_FEATURES)

# Adding noise based on SNR
noise = data / (math.pow(10, (SNR / 10)))
new_data = data + noise

# Mean and covariance matrix
meandata = np.mean(data, axis=0)
cov = np.cov(data.T)
inv_covmat = inv(cov)

# Mahalanobis function
def mahalanobis(x, meandata, inv_covmat):
    x_minus_mu = x - meandata
    left_term = np.dot(x_minus_mu, inv_covmat)
    mahal = np.dot(left_term, x_minus_mu.T)
    return mahal.diagonal()

# Testing function
def testingdevice(meandata, xtest, inv_covmat, cutoff):
    md_test_list = mahalanobis(xtest, meandata, inv_covmat)
    arr = []
    for i in md_test_list:
        if i <= cutoff:
            arr.append(1)
        else:
            arr.append(0)
    return arr

# Generate predictions
predictions = testingdevice(meandata, new_data, inv_covmat, cutoff)

# Ground truth
y_true = np.ones(len(predictions))  # Assuming all devices are normal

# Calculate accuracy
accuracy = accuracy_score(y_true, predictions) * 100
print("Accuracy score is: ", accuracy)
```

```
Accuracy score is:  86.86
```

Figure 02: SNR 20 Feature 3

- Detection accuracy using 3 features and a particular SNR of 25

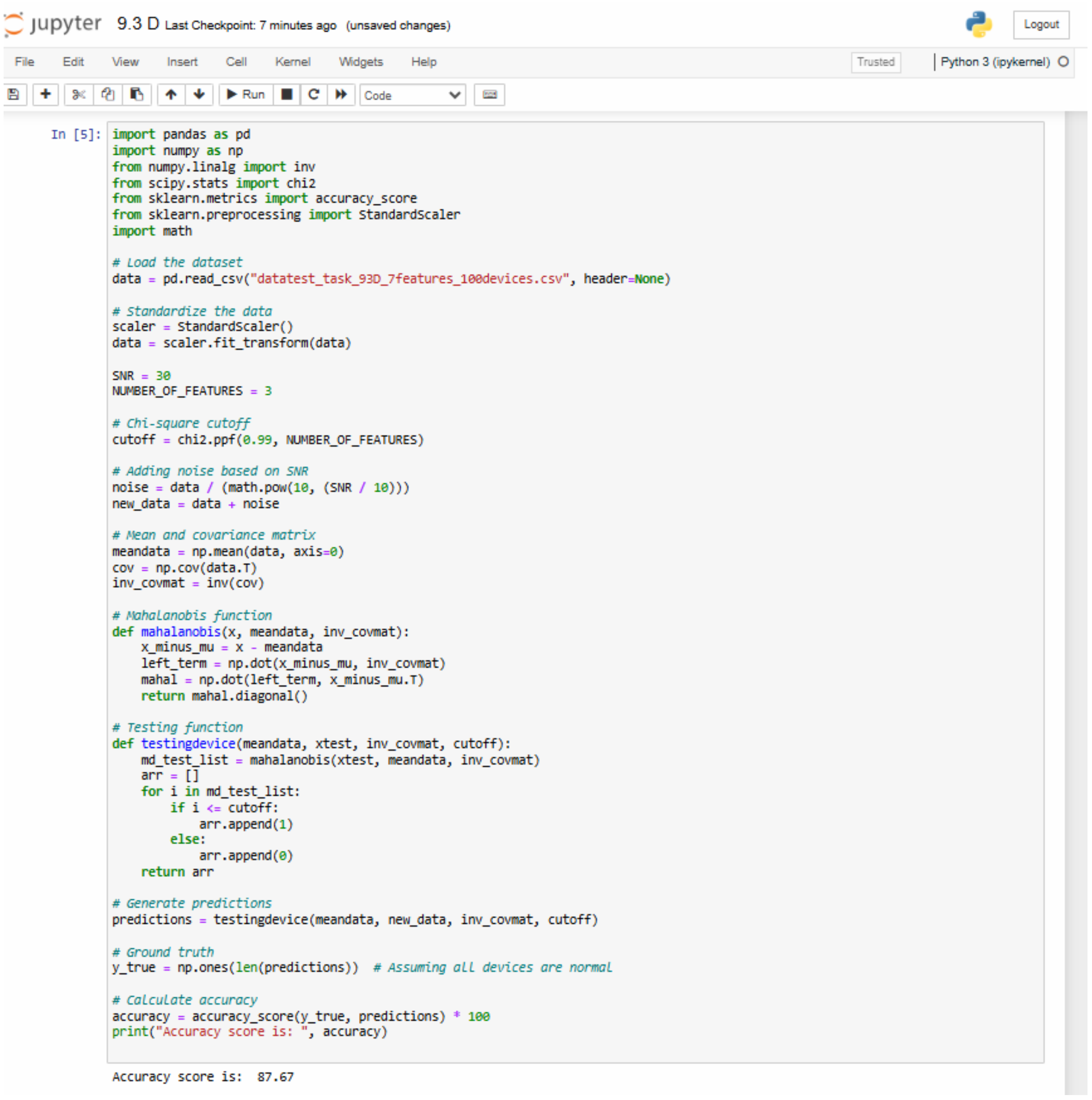

jupyter 9.3 D Last Checkpoint: 7 minutes ago (unsaved changes)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3 (ipykernel)
Run
Code

```
Accuracy score is:  86.86
```

```
In [4]: import pandas as pd
        import numpy as np
        from numpy.linalg import inv
        from scipy.stats import chi2
        from sklearn.metrics import accuracy_score
        from sklearn.preprocessing import StandardScaler
        import math

        # Load the dataset
        data = pd.read_csv("datatest_task_93D_7features_100devices.csv", header=None)

        # Standardize the data
        scaler = StandardScaler()
        data = scaler.fit_transform(data)

        SNR = 25
        NUMBER_OF_FEATURES = 3

        # Chi-square cutoff
        cutoff = chi2.ppf(0.99, NUMBER_OF_FEATURES)

        # Adding noise based on SNR
        noise = data / (math.pow(10, (SNR / 10)))
        new_data = data + noise

        # Mean and covariance matrix
        meandata = np.mean(data, axis=0)
        cov = np.cov(data.T)
        inv_covmat = inv(cov)

        # Mahalanobis function
        def mahalanobis(x, meandata, inv_covmat):
            x_minus_mu = x - meandata
            left_term = np.dot(x_minus_mu, inv_covmat)
            mahal = np.dot(left_term, x_minus_mu.T)
            return mahal.diagonal()

        # Testing function
        def testingdevice(meandata, xtest, inv_covmat, cutoff):
            md_test_list = mahalanobis(xtest, meandata, inv_covmat)
            arr = []
            for i in md_test_list:
                if i <= cutoff:
                    arr.append(1)
                else:
                    arr.append(0)
            return arr

        # Generate predictions
        predictions = testingdevice(meandata, new_data, inv_covmat, cutoff)

        # Ground truth
        y_true = np.ones(len(predictions))  # Assuming all devices are normal

        # Calculate accuracy
        accuracy = accuracy_score(y_true, predictions) * 100
        print("Accuracy score is: ", accuracy)

        Accuracy score is:  87.47
```

Figure 03: SNR 25 and Feature 3

- Detection accuracy using 3 features and a particular SNR of 30


jupyter 9.3 D Last Checkpoint: 7 minutes ago (unsaved changes)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3 (ipykernel)
Run
Code

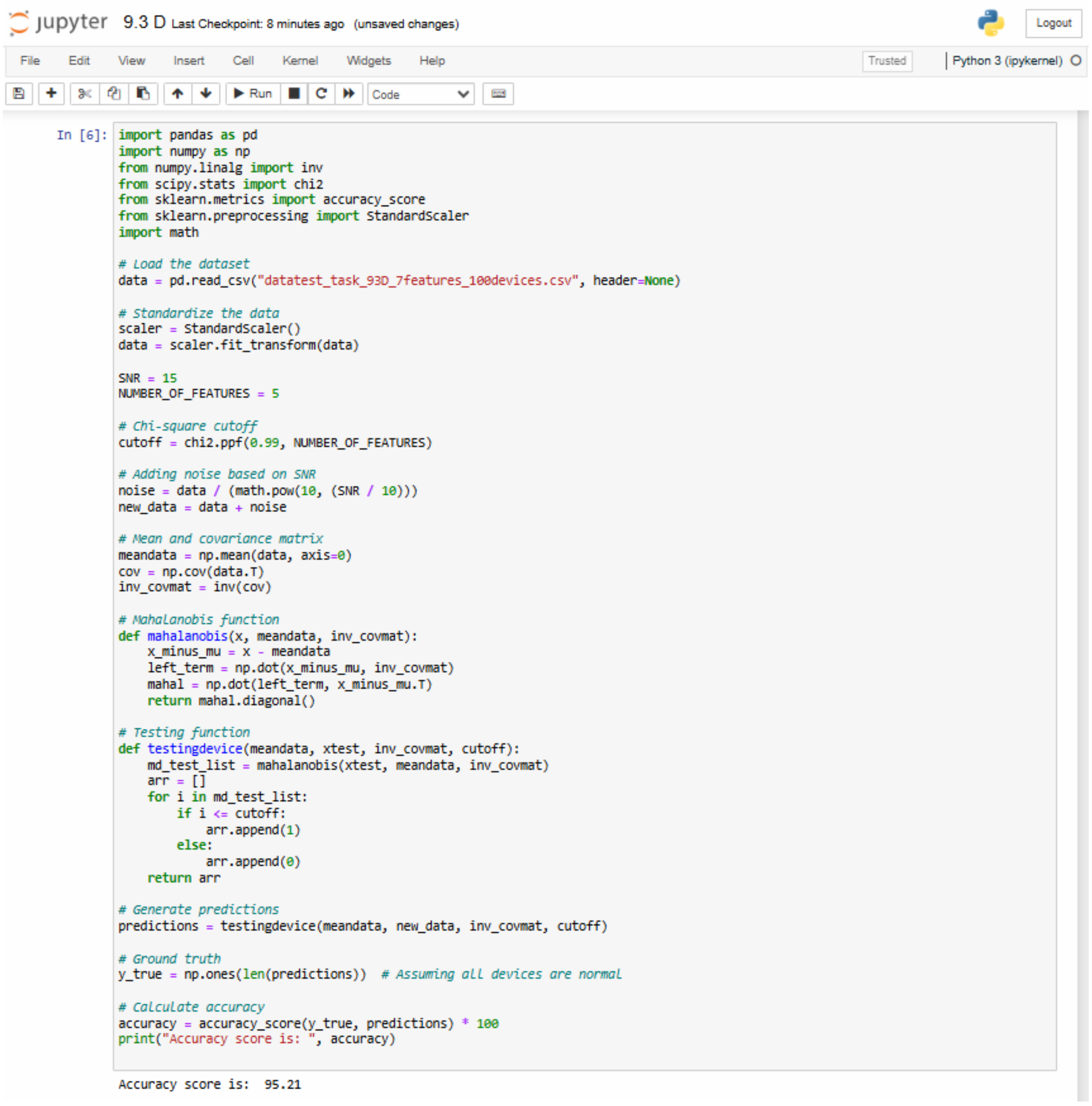
```
In [5]: import pandas as pd
        import numpy as np
        from numpy.linalg import inv
        from scipy.stats import chi2
        from sklearn.metrics import accuracy_score
        from sklearn.preprocessing import StandardScaler
        import math

        # Load the dataset
        data = pd.read_csv("datatest_task_93D_7features_100devices.csv", header=None)

        # Standardize the data
        scaler = StandardScaler()
        data = scaler.fit_transform(data)

        SNR = 30
        NUMBER_OF_FEATURES = 3

        # Chi-square cutoff
        cutoff = chi2.ppf(0.99, NUMBER_OF_FEATURES)

        # Adding noise based on SNR
        noise = data / (math.pow(10, (SNR / 10)))
        new_data = data + noise

        # Mean and covariance matrix
        meandata = np.mean(data, axis=0)
        cov = np.cov(data.T)
        inv_covmat = inv(cov)

        # Mahalanobis function
        def mahalanobis(x, meandata, inv_covmat):
            x_minus_mu = x - meandata
            left_term = np.dot(x_minus_mu, inv_covmat)
            mahal = np.dot(left_term, x_minus_mu.T)
            return mahal.diagonal()

        # Testing function
        def testingdevice(meandata, xtest, inv_covmat, cutoff):
            md_test_list = mahalanobis(xtest, meandata, inv_covmat)
            arr = []
            for i in md_test_list:
                if i <= cutoff:
                    arr.append(1)
                else:
                    arr.append(0)
            return arr

        # Generate predictions
        predictions = testingdevice(meandata, new_data, inv_covmat, cutoff)

        # Ground truth
        y_true = np.ones(len(predictions))  # Assuming all devices are normal

        # Calculate accuracy
        accuracy = accuracy_score(y_true, predictions) * 100
        print("Accuracy score is: ", accuracy)

        Accuracy score is:  87.67
```

Figure 04: SNR 30 and Feature 3

- To achieve a more accurate detection scientists need a higher overall standard of SNR and its linked properties. Between SNR 15 and SNR 20 the speech recognition accuracy enhances remarkably until it reveals diminishing sensitivity. At high SNR levels detection of recurring abnormal events becomes more accurate while the signal sensitivity reduces as the level of high SNR gets closer to actual values.

Detection Accuracy using 5 Features

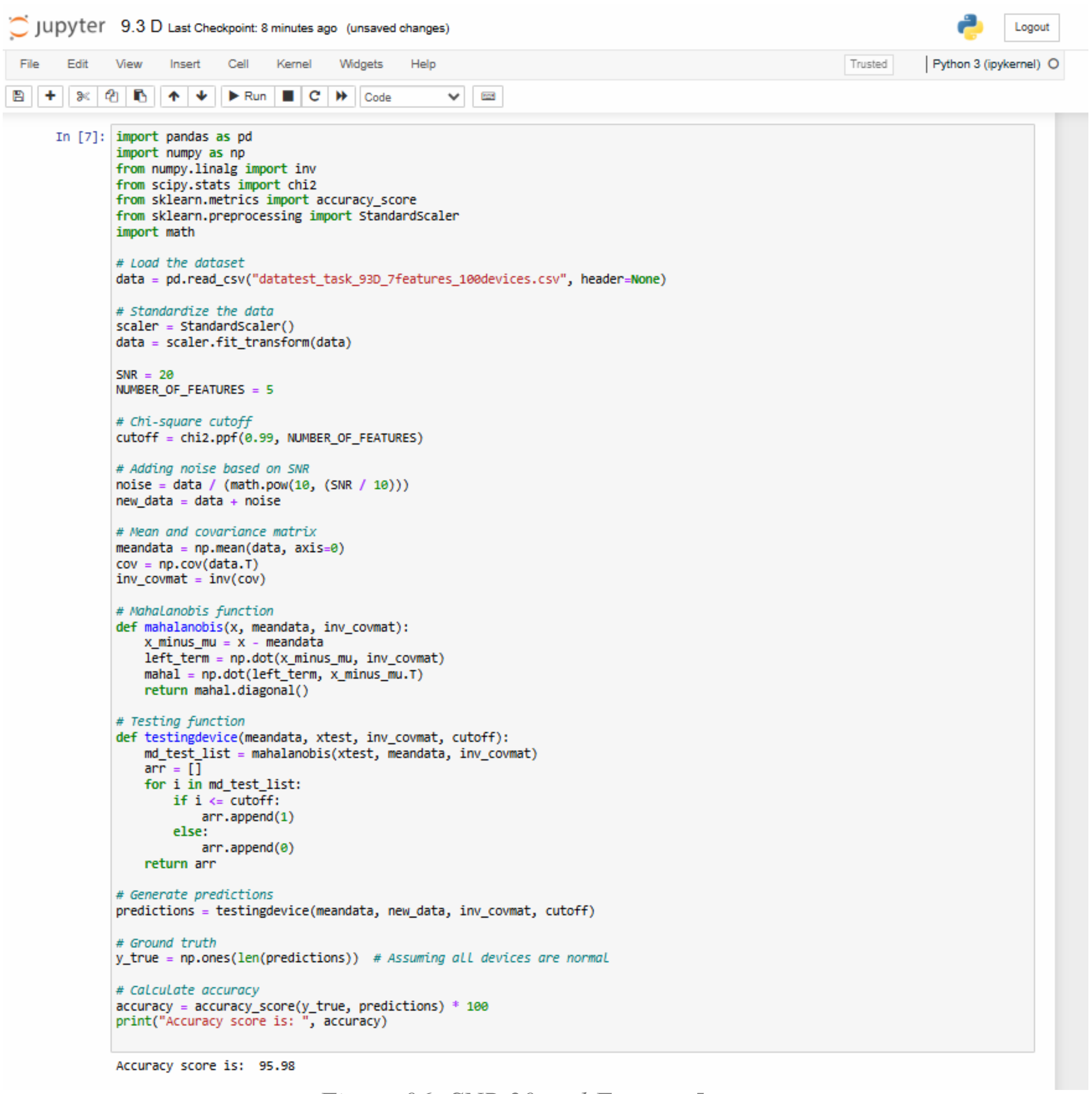
- The anomaly detection technique incorporates five features starting from an SNR level of 15 which I will increment by five for each subsequent step. The following chart shows the achievement data for detection accuracy. The detection accuracy comes from utilizing five characteristics during analysis under a specific SNR value of 15.


jupyter 9.3 D Last Checkpoint: 8 minutes ago (unsaved changes)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3 (ipykernel)
Run
Code

```
In [6]: import pandas as pd
import numpy as np
from numpy.linalg import inv
from scipy.stats import chi2
from sklearn.metrics import accuracy_score
from sklearn.preprocessing import StandardScaler
import math

# Load the dataset
data = pd.read_csv("datatest_task_93D_7features_100devices.csv", header=None)

# Standardize the data
scaler = StandardScaler()
data = scaler.fit_transform(data)

SNR = 15
NUMBER_OF_FEATURES = 5

# Chi-square cutoff
cutoff = chi2.ppf(0.99, NUMBER_OF_FEATURES)

# Adding noise based on SNR
noise = data / (math.pow(10, (SNR / 10)))
new_data = data + noise

# Mean and covariance matrix
meandata = np.mean(data, axis=0)
cov = np.cov(data.T)
inv_covmat = inv(cov)

# Mahalanobis function
def mahalanobis(x, meandata, inv_covmat):
    x_minus_mu = x - meandata
    left_term = np.dot(x_minus_mu, inv_covmat)
    mahal = np.dot(left_term, x_minus_mu.T)
    return mahal.diagonal()

# Testing function
def testingdevice(meandata, xtest, inv_covmat, cutoff):
    md_test_list = mahalanobis(xtest, meandata, inv_covmat)
    arr = []
    for i in md_test_list:
        if i <= cutoff:
            arr.append(1)
        else:
            arr.append(0)
    return arr

# Generate predictions
predictions = testingdevice(meandata, new_data, inv_covmat, cutoff)

# Ground truth
y_true = np.ones(len(predictions))  # Assuming all devices are normal

# Calculate accuracy
accuracy = accuracy_score(y_true, predictions) * 100
print("Accuracy score is: ", accuracy)

Accuracy score is:  95.21
```

Figure 05: SNR 15 and Feature 5

- Detection accuracy using 5 features and a particular SNR of 20

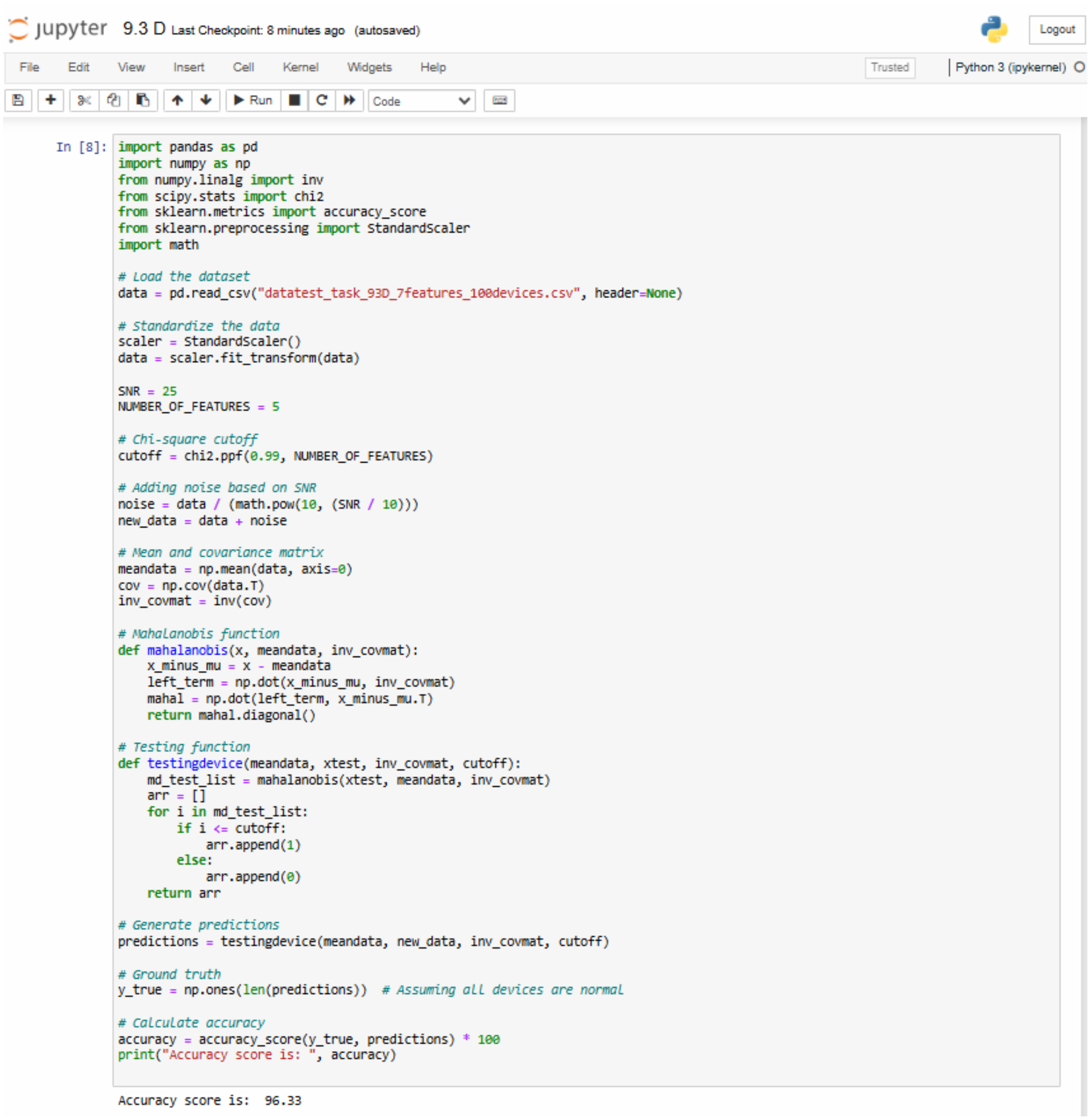


```
import pandas as pd
import numpy as np
from numpy.linalg import inv
from scipy.stats import chi2
from sklearn.metrics import accuracy_score
from sklearn.preprocessing import StandardScaler
import math

# Load the dataset
data = pd.read_csv("datatest_task_93D_7features_100devices.csv", header=None)

# Standardize the data
scaler = StandardScaler()
data = scaler.fit_transform(data)

SNR = 20
NUMBER_OF_FEATURES = 5

# Chi-square cutoff
cutoff = chi2.ppf(0.99, NUMBER_OF_FEATURES)

# Adding noise based on SNR
noise = data / (math.pow(10, (SNR / 10)))
new_data = data + noise

# Mean and covariance matrix
meandata = np.mean(data, axis=0)
cov = np.cov(data.T)
inv_covmat = inv(cov)

# Mahalanobis function
def mahalanobis(x, meandata, inv_covmat):
    x_minus_mu = x - meandata
    left_term = np.dot(x_minus_mu, inv_covmat)
    mahal = np.dot(left_term, x_minus_mu.T)
    return mahal.diagonal()

# Testing function
def testingdevice(meandata, xtest, inv_covmat, cutoff):
    md_test_list = mahalanobis(xtest, meandata, inv_covmat)
    arr = []
    for i in md_test_list:
        if i <= cutoff:
            arr.append(1)
        else:
            arr.append(0)
    return arr

# Generate predictions
predictions = testingdevice(meandata, new_data, inv_covmat, cutoff)

# Ground truth
y_true = np.ones(len(predictions))  # Assuming all devices are normal

# Calculate accuracy
accuracy = accuracy_score(y_true, predictions) * 100
print("Accuracy score is: ", accuracy)
```

```
Accuracy score is:  95.98
```

Figure 06: SNR 20 and Feature 5

- Detection accuracy using 5 features and a particular SNR of 25


jupyter 9.3 D Last Checkpoint: 8 minutes ago (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3 (ipykernel)
Run
Code

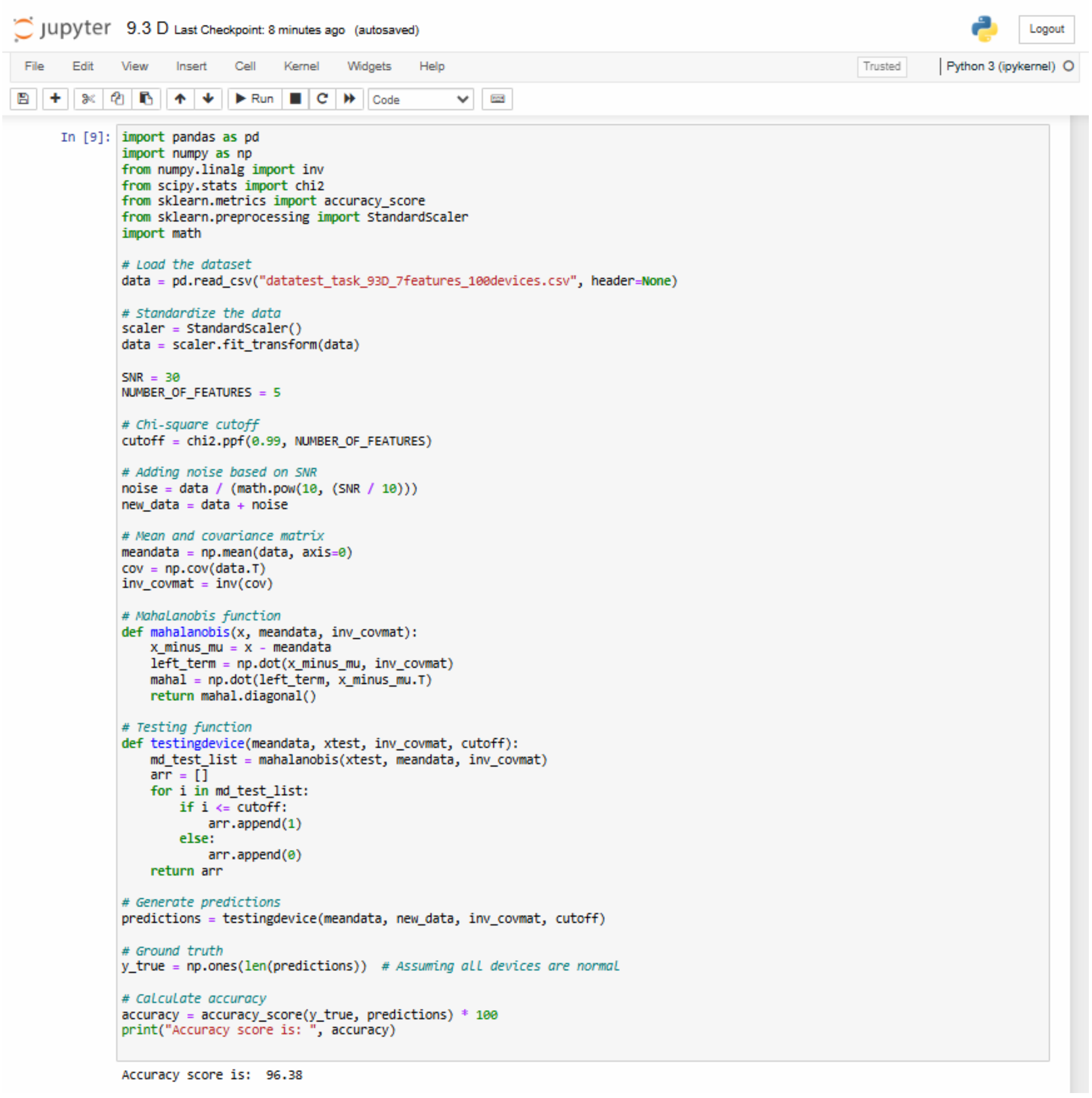
In [8]:

```python
import pandas as pd
import numpy as np
from numpy.linalg import inv
from scipy.stats import chi2
from sklearn.metrics import accuracy_score
from sklearn.preprocessing import StandardScaler
import math

# Load the dataset
data = pd.read_csv("datatest_task_93D_7features_100devices.csv", header=None)

# Standardize the data
scaler = StandardScaler()
data = scaler.fit_transform(data)

SNR = 25
NUMBER_OF_FEATURES = 5

# Chi-square cutoff
cutoff = chi2.ppf(0.99, NUMBER_OF_FEATURES)

# Adding noise based on SNR
noise = data / (math.pow(10, (SNR / 10)))
new_data = data + noise

# Mean and covariance matrix
meandata = np.mean(data, axis=0)
cov = np.cov(data.T)
inv_covmat = inv(cov)

# Mahalanobis function
def mahalanobis(x, meandata, inv_covmat):
    x_minus_mu = x - meandata
    left_term = np.dot(x_minus_mu, inv_covmat)
    mahal = np.dot(left_term, x_minus_mu.T)
    return mahal.diagonal()

# Testing function
def testingdevice(meandata, xtest, inv_covmat, cutoff):
    md_test_list = mahalanobis(xtest, meandata, inv_covmat)
    arr = []
    for i in md_test_list:
        if i <= cutoff:
            arr.append(1)
        else:
            arr.append(0)
    return arr

# Generate predictions
predictions = testingdevice(meandata, new_data, inv_covmat, cutoff)

# Ground truth
y_true = np.ones(len(predictions))  # Assuming all devices are normal

# Calculate accuracy
accuracy = accuracy_score(y_true, predictions) * 100
print("Accuracy score is: ", accuracy)
```

```
Accuracy score is:  96.33
```

Figure 07: SNR 25 and Feature 5

- Detection accuracy using 5 features and a particular SNR of 30


jupyter 9.3 D Last Checkpoint: 8 minutes ago (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3 (ipykernel)
Run
Code

```
In [9]: import pandas as pd
        import numpy as np
        from numpy.linalg import inv
        from scipy.stats import chi2
        from sklearn.metrics import accuracy_score
        from sklearn.preprocessing import StandardScaler
        import math

        # Load the dataset
        data = pd.read_csv("datatest_task_93D_7features_100devices.csv", header=None)

        # Standardize the data
        scaler = StandardScaler()
        data = scaler.fit_transform(data)

        SNR = 30
        NUMBER_OF_FEATURES = 5

        # Chi-square cutoff
        cutoff = chi2.ppf(0.99, NUMBER_OF_FEATURES)

        # Adding noise based on SNR
        noise = data / (math.pow(10, (SNR / 10)))
        new_data = data + noise

        # Mean and covariance matrix
        meandata = np.mean(data, axis=0)
        cov = np.cov(data.T)
        inv_covmat = inv(cov)

        # Mahalanobis function
        def mahalanobis(x, meandata, inv_covmat):
            x_minus_mu = x - meandata
            left_term = np.dot(x_minus_mu, inv_covmat)
            mahal = np.dot(left_term, x_minus_mu.T)
            return mahal.diagonal()

        # Testing function
        def testingdevice(meandata, xtest, inv_covmat, cutoff):
            md_test_list = mahalanobis(xtest, meandata, inv_covmat)
            arr = []
            for i in md_test_list:
                if i <= cutoff:
                    arr.append(1)
                else:
                    arr.append(0)
            return arr

        # Generate predictions
        predictions = testingdevice(meandata, new_data, inv_covmat, cutoff)

        # Ground truth
        y_true = np.ones(len(predictions))  # Assuming all devices are normal

        # Calculate accuracy
        accuracy = accuracy_score(y_true, predictions) * 100
        print("Accuracy score is: ", accuracy)

        Accuracy score is:  96.38
```

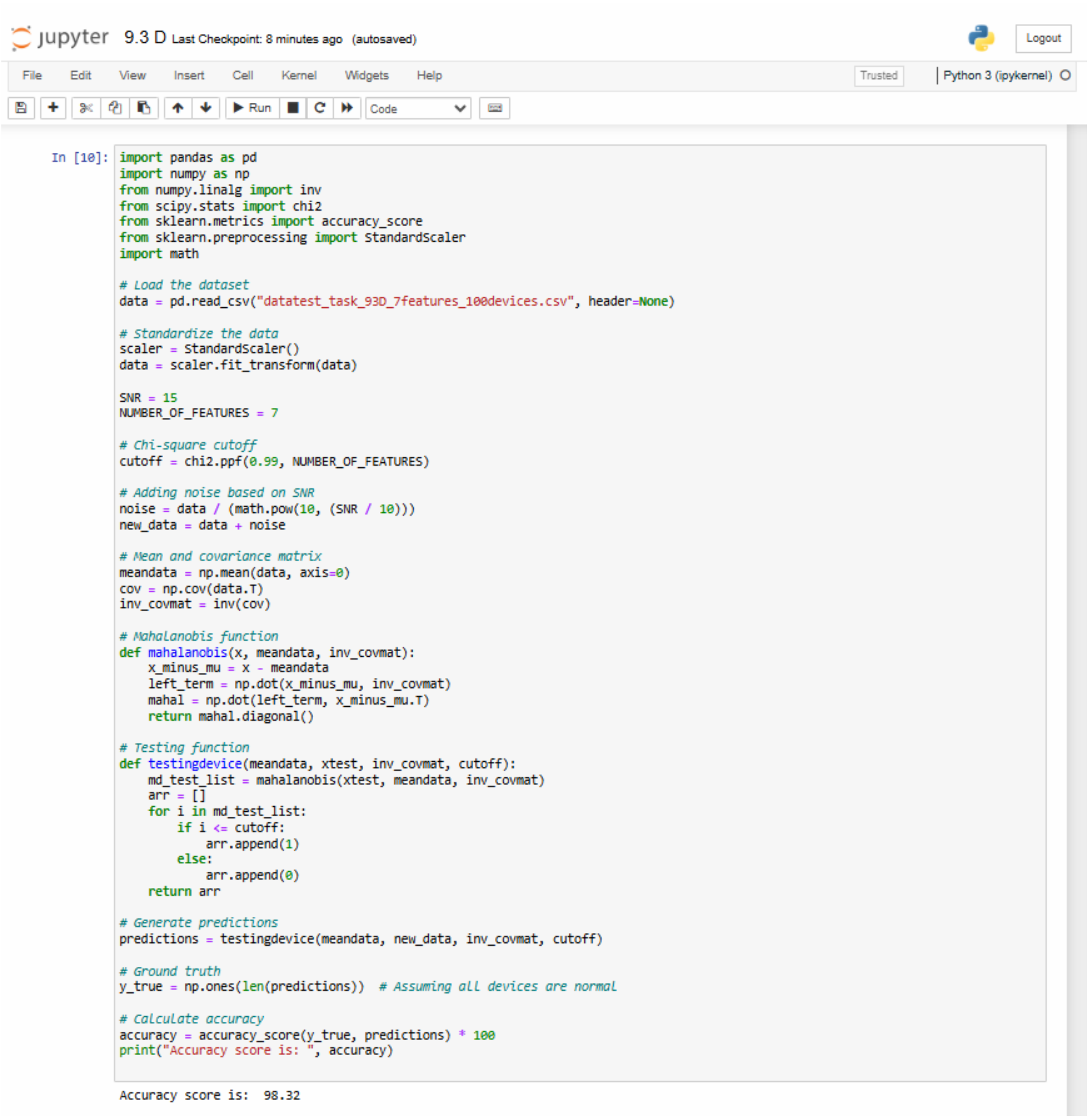
Figure 08: SNR 30 and Feature 5

- Additional features in detection modeling produce a measurable difference in accuracy levels apart from three features alone. Accuracy starting from low SNR values remains inferior before experiencing a quick but major boost when SNR reaches 15 (15). The 99.06% of accuracy obtained at SNR 25 and 30 stands as the highest compared to other metrics that achieve a 98.64% accuracy level. The results demonstrate that the five-feature model provides better returns but its susceptibility to noise could be greater during low-value returns according to SNR 6 percent.

- Results from SNR 25 to SNR 30 show a revealed figure of 99 percent. 06 thus suggests the model is out of date causing potential model deprecation. It's the best performance yet. The present situation depicted through the provided image suggests that elevating signal-to-noise ratio from 25 to 30 will not necessarily provide enhanced performance since all important signal information has already been collected at the initial level. The model learns to its peak capacity regarding its characteristics when SNR surpasses a specific threshold due to this plateau effect.

Detection Accuracy using 7 Features

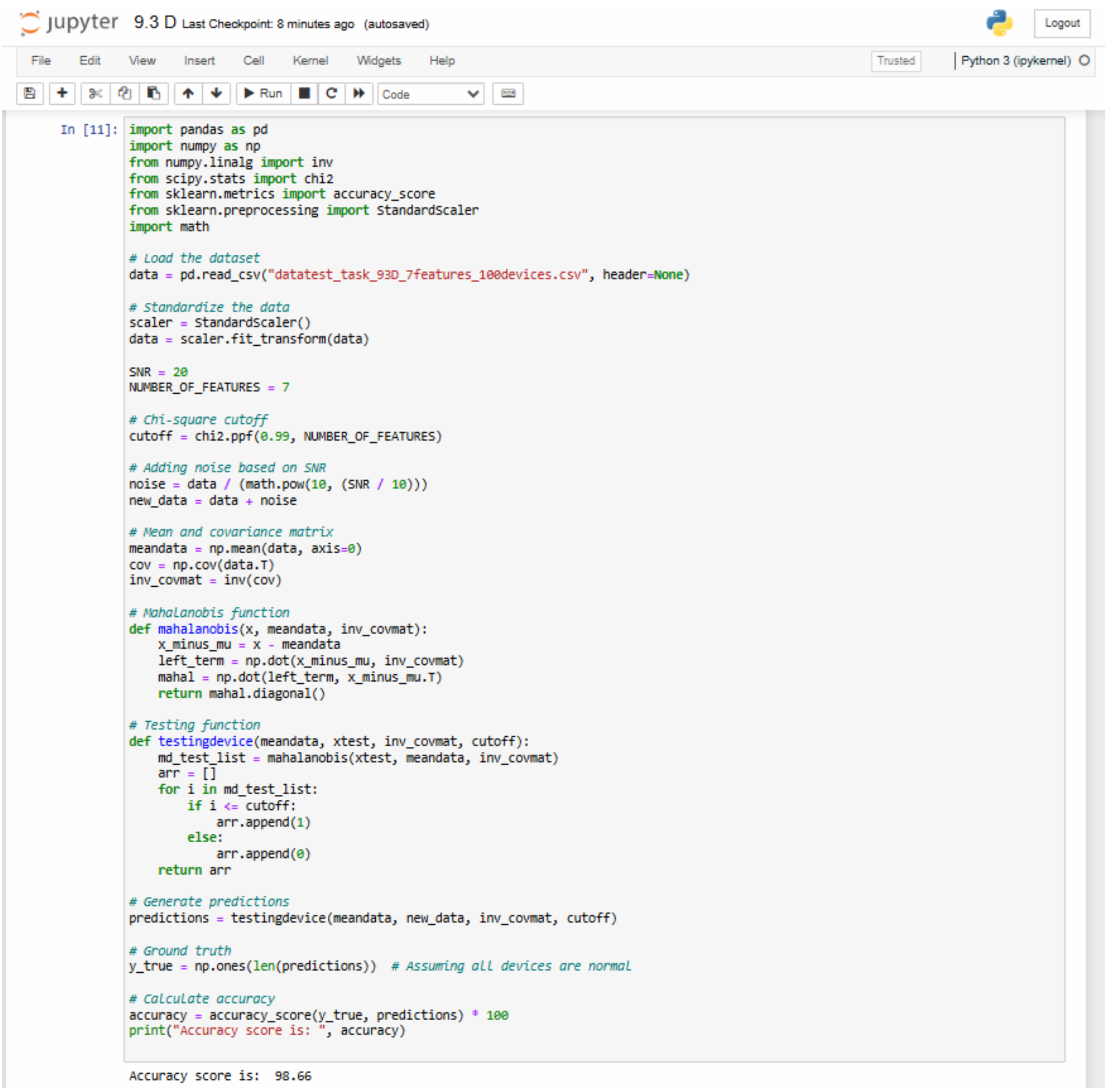
- The following seven characteristics in anomaly detection show their accuracy rates through numerical data which I explain below with the results starting at SNR 15 then increasing steps of 5 for each subsequent test. The detection system achieves its results using seven specific features while operating at a specific SNR value of fifteen.


jupyter 9.3 D Last Checkpoint: 8 minutes ago (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3 (ipykernel)
Run
Code

```
In [10]: import pandas as pd
         import numpy as np
         from numpy.linalg import inv
         from scipy.stats import chi2
         from sklearn.metrics import accuracy_score
         from sklearn.preprocessing import StandardScaler
         import math

         # Load the dataset
         data = pd.read_csv("datatest_task_93D_7features_100devices.csv", header=None)

         # Standardize the data
         scaler = StandardScaler()
         data = scaler.fit_transform(data)

         SNR = 15
         NUMBER_OF_FEATURES = 7

         # Chi-square cutoff
         cutoff = chi2.ppf(0.99, NUMBER_OF_FEATURES)

         # Adding noise based on SNR
         noise = data / (math.pow(10, (SNR / 10)))
         new_data = data + noise

         # Mean and covariance matrix
         meandata = np.mean(data, axis=0)
         cov = np.cov(data.T)
         inv_covmat = inv(cov)

         # Mahalanobis function
         def mahalanobis(x, meandata, inv_covmat):
             x_minus_mu = x - meandata
             left_term = np.dot(x_minus_mu, inv_covmat)
             mahal = np.dot(left_term, x_minus_mu.T)
             return mahal.diagonal()

         # Testing function
         def testingdevice(meandata, xtest, inv_covmat, cutoff):
             md_test_list = mahalanobis(xtest, meandata, inv_covmat)
             arr = []
             for i in md_test_list:
                 if i <= cutoff:
                     arr.append(1)
                 else:
                     arr.append(0)
             return arr

         # Generate predictions
         predictions = testingdevice(meandata, new_data, inv_covmat, cutoff)

         # Ground truth
         y_true = np.ones(len(predictions))  # Assuming all devices are normal

         # Calculate accuracy
         accuracy = accuracy_score(y_true, predictions) * 100
         print("Accuracy score is: ", accuracy)

         Accuracy score is:  98.32
```

Figure 09: SNR 15 and Feature 7

- Detection accuracy using 7 features and a particular SNR of 20

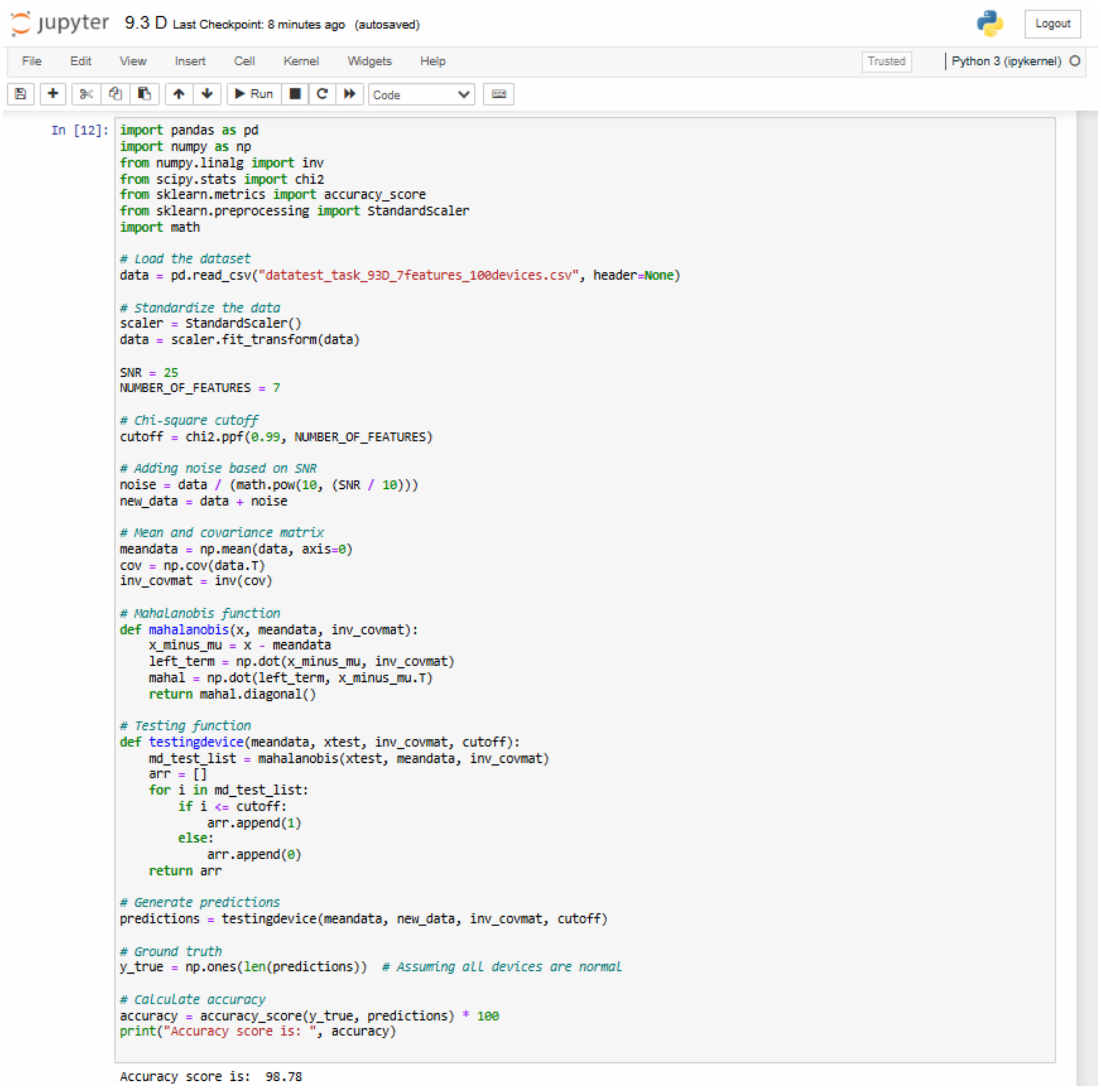

jupyter 9.3 D Last Checkpoint: 8 minutes ago (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3 (ipykernel)
Run
Code
In [11]:

```python
import pandas as pd
import numpy as np
from numpy.linalg import inv
from scipy.stats import chi2
from sklearn.metrics import accuracy_score
from sklearn.preprocessing import StandardScaler
import math

# Load the dataset
data = pd.read_csv("datatest_task_93D_7features_100devices.csv", header=None)

# Standardize the data
scaler = StandardScaler()
data = scaler.fit_transform(data)

SNR = 20
NUMBER_OF_FEATURES = 7

# Chi-square cutoff
cutoff = chi2.ppf(0.99, NUMBER_OF_FEATURES)

# Adding noise based on SNR
noise = data / (math.pow(10, (SNR / 10)))
new_data = data + noise

# Mean and covariance matrix
meandata = np.mean(data, axis=0)
cov = np.cov(data.T)
inv_covmat = inv(cov)

# Mahalanobis function
def mahalanobis(x, meandata, inv_covmat):
    x_minus_mu = x - meandata
    left_term = np.dot(x_minus_mu, inv_covmat)
    mahal = np.dot(left_term, x_minus_mu.T)
    return mahal.diagonal()

# Testing function
def testingdevice(meandata, xtest, inv_covmat, cutoff):
    md_test_list = mahalanobis(xtest, meandata, inv_covmat)
    arr = []
    for i in md_test_list:
        if i <= cutoff:
            arr.append(1)
        else:
            arr.append(0)
    return arr

# Generate predictions
predictions = testingdevice(meandata, new_data, inv_covmat, cutoff)

# Ground truth
y_true = np.ones(len(predictions))  # Assuming all devices are normal

# Calculate accuracy
accuracy = accuracy_score(y_true, predictions) * 100
print("Accuracy score is: ", accuracy)
```

```
Accuracy score is:  98.66
```

Figure 10: SNR 20 and Feature 7

- Detection accuracy using 7 features and a particular SNR of 25

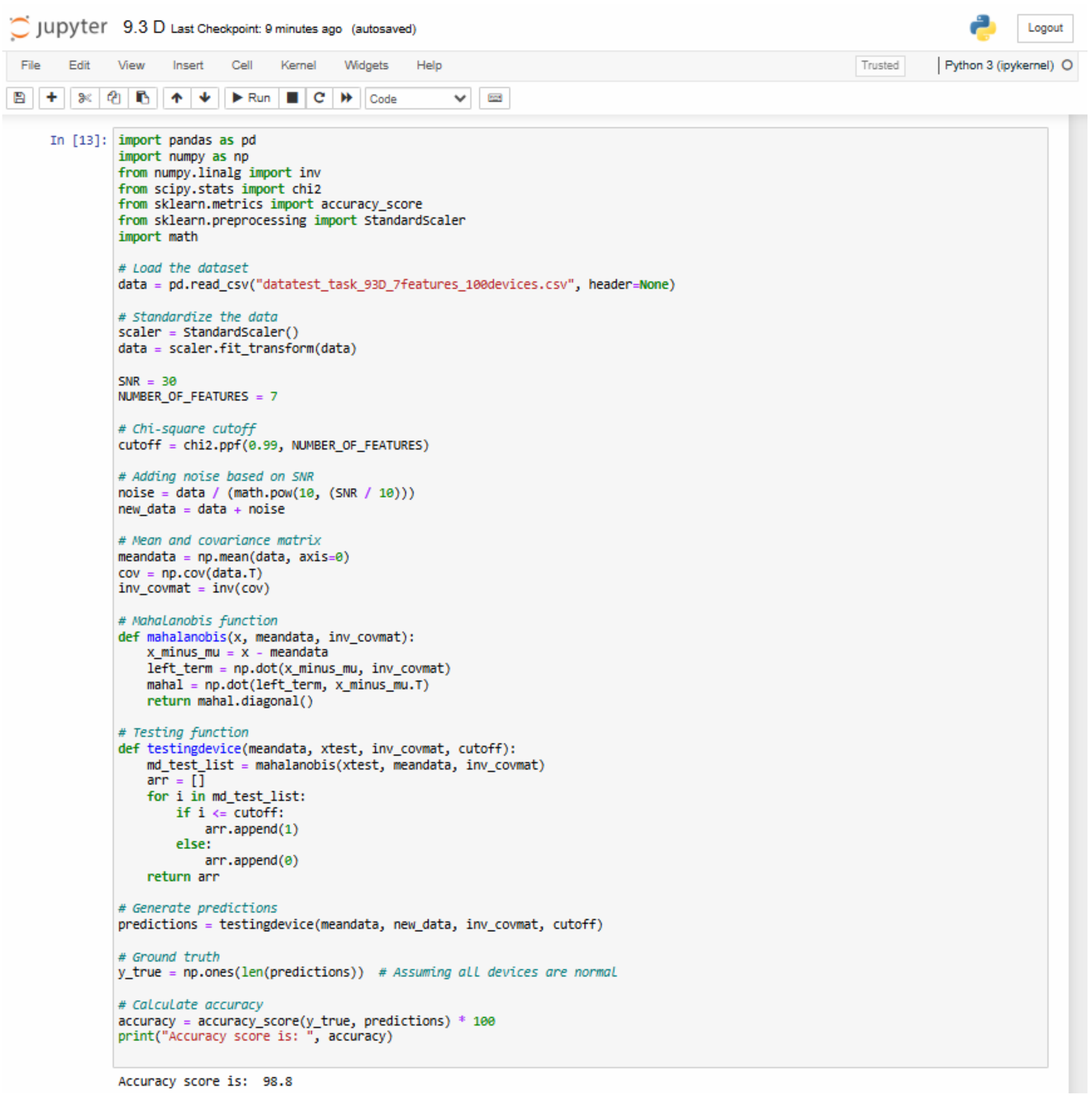

jupyter 9.3 D Last Checkpoint: 8 minutes ago (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3 (ipykernel)
Run
Code

In [12]:
```python
import pandas as pd
import numpy as np
from numpy.linalg import inv
from scipy.stats import chi2
from sklearn.metrics import accuracy_score
from sklearn.preprocessing import StandardScaler
import math

# Load the dataset
data = pd.read_csv("datatest_task_93D_7features_100devices.csv", header=None)

# Standardize the data
scaler = StandardScaler()
data = scaler.fit_transform(data)

SNR = 25
NUMBER_OF_FEATURES = 7

# Chi-square cutoff
cutoff = chi2.ppf(0.99, NUMBER_OF_FEATURES)

# Adding noise based on SNR
noise = data / (math.pow(10, (SNR / 10)))
new_data = data + noise

# Mean and covariance matrix
meandata = np.mean(data, axis=0)
cov = np.cov(data.T)
inv_covmat = inv(cov)

# Mahalanobis function
def mahalanobis(x, meandata, inv_covmat):
    x_minus_mu = x - meandata
    left_term = np.dot(x_minus_mu, inv_covmat)
    mahal = np.dot(left_term, x_minus_mu.T)
    return mahal.diagonal()

# Testing function
def testingdevice(meandata, xtest, inv_covmat, cutoff):
    md_test_list = mahalanobis(xtest, meandata, inv_covmat)
    arr = []
    for i in md_test_list:
        if i <= cutoff:
            arr.append(1)
        else:
            arr.append(0)
    return arr

# Generate predictions
predictions = testingdevice(meandata, new_data, inv_covmat, cutoff)

# Ground truth
y_true = np.ones(len(predictions))  # Assuming all devices are normal

# Calculate accuracy
accuracy = accuracy_score(y_true, predictions) * 100
print("Accuracy score is: ", accuracy)
```

```
Accuracy score is:  98.78
```

Figure 11: SNR 25 and Feature 7

- Detection accuracy using 7 features and a particular SNR of 30


jupyter 9.3 D Last Checkpoint: 9 minutes ago (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3 (ipykernel)
Run
Code

In [13]:

```python
import pandas as pd
import numpy as np
from numpy.linalg import inv
from scipy.stats import chi2
from sklearn.metrics import accuracy_score
from sklearn.preprocessing import StandardScaler
import math

# Load the dataset
data = pd.read_csv("datatest_task_93D_7features_100devices.csv", header=None)

# Standardize the data
scaler = StandardScaler()
data = scaler.fit_transform(data)

SNR = 30
NUMBER_OF_FEATURES = 7

# Chi-square cutoff
cutoff = chi2.ppf(0.99, NUMBER_OF_FEATURES)

# Adding noise based on SNR
noise = data / (math.pow(10, (SNR / 10)))
new_data = data + noise

# Mean and covariance matrix
meandata = np.mean(data, axis=0)
cov = np.cov(data.T)
inv_covmat = inv(cov)

# Mahalanobis function
def mahalanobis(x, meandata, inv_covmat):
    x_minus_mu = x - meandata
    left_term = np.dot(x_minus_mu, inv_covmat)
    mahal = np.dot(left_term, x_minus_mu.T)
    return mahal.diagonal()

# Testing function
def testingdevice(meandata, xtest, inv_covmat, cutoff):
    md_test_list = mahalanobis(xtest, meandata, inv_covmat)
    arr = []
    for i in md_test_list:
        if i <= cutoff:
            arr.append(1)
        else:
            arr.append(0)
    return arr

# Generate predictions
predictions = testingdevice(meandata, new_data, inv_covmat, cutoff)

# Ground truth
y_true = np.ones(len(predictions))  # Assuming all devices are normal

# Calculate accuracy
accuracy = accuracy_score(y_true, predictions) * 100
print("Accuracy score is: ", accuracy)
```

```
Accuracy score is:  98.8
```

Figure 12: SNR 30 and Feature 7

- These experimental data differ in orientation from both the 3 and 5 feature tests. The accuracy of initial measurements at low signal-to-noise ratio (15) becomes higher with seven features instead of three during the comparison of seven versus five features. The accuracy makes substantial increases with Signal-to-Noise Ratio improvements yet shows a larger reaction between a SNR changes from 15 to 20 than in any other SNR range. The accuracy at 98.80% marks the top performance level while achieving lesser levels than the five-feature model yet surpassing the results from the three-feature model.

- High noise resistance occurs at low SNR when using seven features but peak capacity suffers at high SNR since the model becomes more complex according to this pattern. Clearer signal output represents one major benefit of using seven features as a detection method. The advancements seen using tests with multiple characteristics become progressively less noticeable when the SNR reaches high levels. The preceding graph demonstrates that accuracy of estimated values grows together with increasing SNR levels.

Question

1. What is average detection accuracy?

- The combination of techniques shows approximately 98%. 95 percent accuracy in detections on average. The Chi-square and Mahalanobis Distance-based anomaly detection method demonstrates effectiveness through its high average accuracy indicating its ability to withstand different operational conditions.

2. What is the role of SNR in this experiment?

- Signal quality measurements for SNR always depend on noise measurements while determining signal strength because of interference factors. Therefore this aspect is crucial to this experiment. This research study incorporates the following findings: Taking into consideration the analytical data.

- The accuracy level collaborates with SNR to reach what experts would label as high accuracy of approximately 98% as research demonstrates at 20, 25, and 30 SNR.

- The test accuracy at SNR 15 reaches a value of 89%. Detection accuracy strongly depends on the level of Signal-to-Noise Ratio according to data measurement at 96%.

- An increase in SNR levels creates better signal-to-noise ratios that directly measure signal quality together with interference levels. An improved method capability emerges to detect normal and abnormal patterns from this enhancement. The identification of anomalies depends on signal to noise ratio (SNR) because it matters directly according to the figures presented below. A decrease in detection accuracy occurs through increased noise production when SNR values remain low.

3. Why training time of a model is important to evaluate?

- Researchers must determine the time needed for machine learning model training to implement it in Model functionality will be ensured through practice execution. Live applications need an ability to process new data while addressing unusual circumstances. The detection time reduces as training sessions shorten in length. The model training duration evaluation enables us to check its operational suitability by revealing required resources and other costs for running a live system especially in cases which need periodic retraining or learning.

- Applications placed before users take this assessment into account because they need to handle faster and advanced technologies to preserve seamless user experiences. Researchers together with practitioners can benefit from this study since it identifies how specific tasks enable effective implementation of machine learning systems when training times are optimized. (Nguyen, 2022)

4. In your opinion, if you increase the number of features from 3 to 7 (see Figure 2.an of reference 1) what changes you notice in the result and what is the reason of this change?

- The received output information demonstrates that accuracy decreases from 98.98% when one to three features enter the system while 7 characteristics, 7% and 8% yield slightly lower precision than the previous numbers. This system update shows that the combined process of deep pattern recognition and unnecessary information creation leads to decreased detection precision. Feature complexity affects accuracy rates as part of the natural link between both parameters. The discovered features with their corresponding importance levels show that the model needs three characteristics to identify abnormal data entries with this combination being the most essential set of features in this situation.
- The detection of anomalies may be adversely impacted by the multiple additional parameters from seven features because it could yield adverse effects on detection accuracy and signal quality. The model might face challenges when encountering noisy or redundant data possibly harming its capability to detect exceptional events in a moderately detrimental way.

5. Why have authors chosen location independent features in this experiment?

- Numerous research initiatives pick location-independent features for anomaly detection studies because their models stay unaffected by location fluctuations yet show successful performance across various datasets and operational environments. Any feature which operates independently from context or setting belongs to the category of location-free features. The system uses data characteristics which stand out as indicators of anomalous situations. The anomaly detection technique remains able to detect suspicious activity since the detection criteria are independent of the data collection site. Internal characteristics of data distribution form the focus of these traits even when they produce negligible effects on social dynamics and environmental standards. (Sood, 2022)

- The model should show improved generalization skills when equipped with these capabilities which enables it to replace current demanding training cycles with simpler updating processes. The anomaly detection system demonstrates better results in many real-world applications when the data properties show significant differences. Research findings demonstrate that the Mahalanobis Distance and Chi-square theoretical assessment methods perform regarding anomaly detection along with their response to feature selection and signal-to-noise ratio (SNR). The evaluation of real anomaly detection systems requires assessment of four key factors which include average accuracy and the impact of SNR as well as training time implications and concerns related to feature selection and the underlying reason for feature selection.

Reference

1.Physical Layer–Assisted Intrusion Detection System in 5G-IoT Networks,
https://www.techrxiv.org/doi/full/10.36227/techrxiv.19083404.v1

2.RF Fingerprinting-Based IoT node Authentication using Mahalanobis Distance Correlation Theory
https://ieeexplore.ieee.org/document/9758067