FAKULTI TEKNOLOGI MAKLUMAT DAN KOMUNIKASI

SEMESTER 1 2023/2024

ARTIFICIAL INTELLIGENCE PROJECT MANAGEMENT

(BITI 3533)

BITI

PROJECT TITLE:

PREDICTING AUTOMOTIVE VEHICLES ENGINE HEALTH

PREPARED BY:

Name	Matric Number
KEE CHIN YEW	B032110144
MUHAMMAD FAKHRUL HAZWAN	B032110357
NABIL SYAZANI	B032110353
SAIFUL ADLI	B032110275

TABLE OF CONTENTS

Project Charter

<table>
<tr><td>Project Name</td><td colspan="3">Predicting Automotive Vehicles Engine Health</td></tr>
<tr><td>Project Sponsor</td><td>Prof. Ts. Dr. Buhanuddin Bin Mohd Aboobaider</td><td>Project Manager</td><td>Kee Chin Yew</td></tr>
<tr><td>Date of project approval</td><td>28th May 2023</td><td>Last Revision Date</td><td>29 May 2023</td></tr>
<tr><td>Project Description</td><td colspan="3">Classify engine conditions and recommend maintenance based on various engine parameters.</td></tr>
<tr><td>Scope</td><td colspan="3">Implementation of Artificial Neural Network (ANN) aimed to predict and classify the health engine condition of automotive engines with a focus on optimizing maintenance recommendations for optimal vehicle performance.</td></tr>
<tr><td>Business Case</td><td colspan="3">The critical need for predictive maintenance in the automotive industry, with the goal of reducing the risk of engine failures, associated repair costs, and potential accidents. This project aims to provide a cost-effective solution for improving engine health assessment, optimizing maintenance schedules, and ultimately enhancing the overall performance and safety of automotive vehicles.</td></tr>
<tr><td rowspan="4">Constraint (in priority order)</td><td>Time</td><td colspan="2">4 months</td></tr>
<tr><td>Budget</td><td colspan="2">4 developers</td></tr>
<tr><td>Scope</td><td colspan="2">To be decide</td></tr>
<tr><td>Quality</td><td colspan="2">Prioritize time and budget over quality</td></tr>
<tr><td>Project Deliverables</td><td colspan="3">Deliverables encompass a functional predictive maintenance system for heavy vehicle brake pads, emphasizing safety, reduced downtime, and cost efficiency. Documentation, user training, integration capabilities, performance reports, and a cost-benefit analysis contribute to a comprehensive and effective solution.</td></tr>
<tr><td>Benefits</td><td colspan="3">Cost savings, enhanced safety through early issue detection, improved operational efficiency with data-driven decision-making, better vehicle performance, reduced downtime, and enhanced fleet management.</td></tr>
<tr><td rowspan="3">Steering communities</td><td>Project Leader</td><td rowspan="3">Project Team</td><td>Developers</td></tr>
<tr><td>Finance Leader</td><td></td></tr>
<tr><td>Senior Manager</td><td></td></tr>
<tr><td rowspan="2">Key Stakeholders</td><td rowspan="2">Project Sponsor</td><td>Project Manager</td><td>Maintenance Personnel</td></tr>
<tr><td colspan="2">Data Scientists</td></tr>
<tr><td>Risks</td><td colspan="3">Potential inaccuracies in the predictive maintenance model. Inaccuracies in the ANN model predicting engine health could result in faulty maintenance recommendations, jeopardizing the safety and reliability of the vehicle fleet.</td></tr>
</table>

INTRODUCTION

This project focuses on automotive engine health. Engines in vehicles are among the most crucial components. Similar to a car's brain, the engine provides the power required for a vehicle to run. Thus, to ensure optimal vehicle performance, regular engine maintenance is required. The engine health of automobiles enables us to assess the engine's condition and determine whether maintenance is necessary.

Car engines are complex, with many parts that need attention to avoid any permanent damage. Most cars run on internal combustion engines, where fuel ignites and burns inside the engine, turning that energy into the power needed to move the car. The engine consists of a fixed cylinder and a moving piston, working together to spin the crankshaft when combustion gases push the piston. This motion is what moves the car's wheels.

The engine's condition cannot be determined directly due to the design and construction of automobile engines. Even though some of the engines are operating efficiently right now, there may be hidden and major risks associated with them. Inaccurate estimation of engine conditions may cause accidents in the future. However, there are a few characteristics that let us predict the engine's state. The work done by the engine, the coolant in the engine and the lubricant oil of the engine can be important variables for determining the condition of the engine because these features show the performance of the engine.

The main contributions of this research work include:

- Address the main issue which is the prediction of engine condition.
- Develop an Artificial Neural Network (ANN) method to classify the condition of the engine to predict the requirement of maintenance of the engine.
- Using Manual Grid Search to find out the best parameter for the ANN model to obtain the most accurate prediction.

DETAIL ABOUT SOFTWARE USED

The software used to create the automotive engine health model is PyCharm. PyCharm is an Integrated Development Environment (IDE) designed for Python development. PyCharm is packed with tools for coding, testing, and fixing Python projects, making it great for building models to predict automotive engine health. It also supports Jupyter Notebooks and has scientific packages, which are helpful for our project to make better predictions about engine health and do exploratory data analysis.

The availability of libraries in PyCharm is another key factor in choosing the software for this project. PyCharm includes Scikit-learn, a Python library for machine learning that offers effective tools for analyzing data and building models. This is especially useful for our project focused on engine health prediction. Moreover, PyCharm also supports TensorFlow, a deep learning framework that can be seamlessly incorporated into our Engine Health prediction model. These two libraries will be used in the project.

Microsoft Excel is integrated into the software toolkit to bolster data manipulation and analysis tasks. Leveraging Excel's functionalities, the project benefits from efficient handling and organization of data, contributing to the accuracy of the engine health predictions.

Microsoft Word plays a vital role in documentation within the project. It is utilized for creating and maintaining project reports, analysis summaries, and other essential documentation. Microsoft Word seamlessly integrates into the workflow, ensuring clarity and effective communication throughout the project.

DETAIL ON AI PROJECT MANAGEMENT

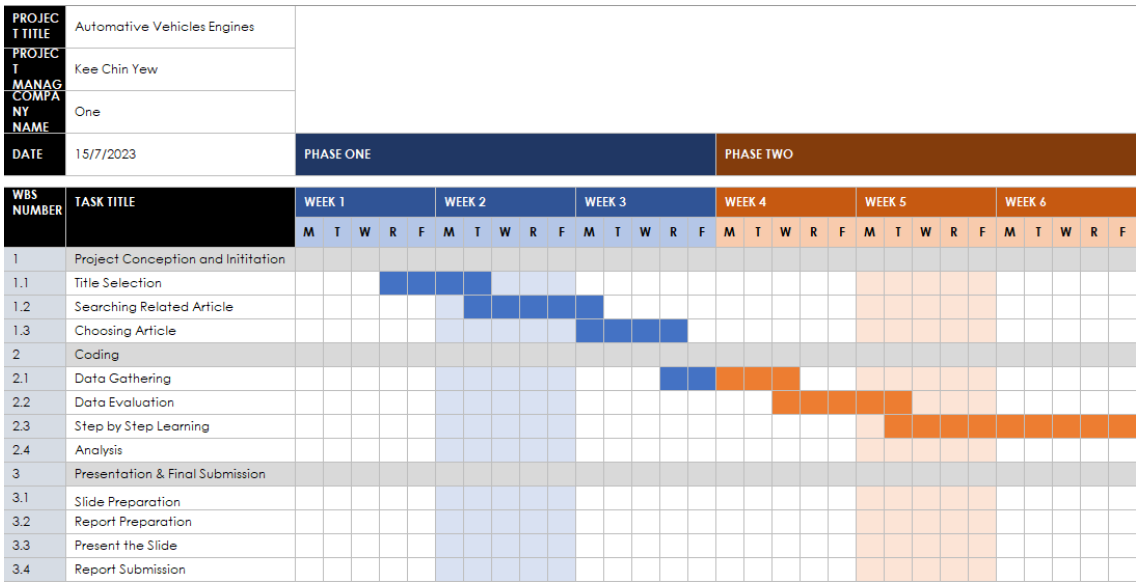
The Work Breakdown Structure below consists of 3 main parts which is Project Conception & Initiation, Coding and Presentation & Final Submission. The total duration of the project is 12 weeks.

PROJECT TITLE	Automative Vehicles Engines
PROJECT MANAGER	Kee Chin Yew
COMPANY NAME	One
DATE	15/7/2023

WBS NUMBER	TASK TITLE	PARTICIPANTS	START DATE	DUE DATE	DURATION	% of TASK COMPLETE
1	Project Conception and Inititation					
1.1	Title Selection	KEE C Y, FAKHRUL H, NABIL S	1/6/2023	4/6/2023	3	100%
1.2	Searching Related Article	KEE C Y, FAKHRUL H, NABIL S	4/6/2023	8/6/2023	4	100%
1.3	Choosing Article	KEE C Y, FAKHRUL H, NABIL S	8/6/2023	11/6/2023	3	100%
2	Coding					
2.1	Data Gathering	NABIL S	11/6/2023	15/6/2023	4	100%
2.2	Data Evaluation	SAIFUL A	15/6/2023	19/6/2023	4	100%
2.3	Step by Step Learning	FAKHRUL H	19/6/2023	3/6/2023	14	100%
2.4	Analysis	KEE C Y	3/7/2023	14/7/2023	11	100%
3	Presentation & Final Submission					
3.1	Slide Preparation	KEE C Y, FAKHRUL H, NABIL S	14/7/2023	20/7/2023	6	80%
3.2	Report Preparation	KEE C Y, FAKHRUL H, NABIL S	14/7/2023	25/7/2023	11	50%
3.3	Present the Slide	KEE C Y, FAKHRUL H, NABIL S	21/7/2023	21/7/2023	0	0%
3.4	Report Submission	KEE C Y	26/7/2023	26/7/2023	0	0%

WBS Chart

Phase one and Phase two included two main parts which are Project Conception & Initiation and Coding. These two phases took a total of six weeks.



WBS Phase One & Two

Phase three and Phase four included two main parts which are Coding and Presentation & Final Submission. These two phases took a total of six weeks.

PROJECT TITLE	Automative Vehicles Engines
PROJECT MANAG	Kee Chin Yew
COMPANY NAME	One
DATE	15/7/2023

WBS NUMBER	TASK TITLE	WEEK 7					WEEK 8					WEEK 9					WEEK 10					WEEK 11					WEEK 12				
		M	T	W	R	F	M	T	W	R	F	M	T	W	R	F	M	T	W	R	F	M	T	W	R	F	M	T	W	R	F
1	Project Conception and Initiation																														
1.1	Title Selection																														
1.2	Searching Related Article																														
1.3	Choosing Article																														
2	Coding																														
2.1	Data Gathering																														
2.2	Data Evaluation																														
2.3	Step by Step Learning																														
2.4	Analysis																														
3	Presentation & Final Submission																														
3.1	Slide Preparation																														
3.2	Report Preparation																														
3.3	Present the Slide																														
3.4	Report Submission																														

WBS Phase Three & Four

The total project cost estimate is RM258,225.60. The largest category of costs is training and support, which is estimated to be RM87,050.00. The next largest category of costs is project management, which is estimated to be RM66,758.00. The cost of hardware is estimated to be RM5,000.00, and the cost of software is estimated to be RM50,800.00. Testing is estimated to cost RM5,580.00 and reserves are estimated to be RM43,037.60.

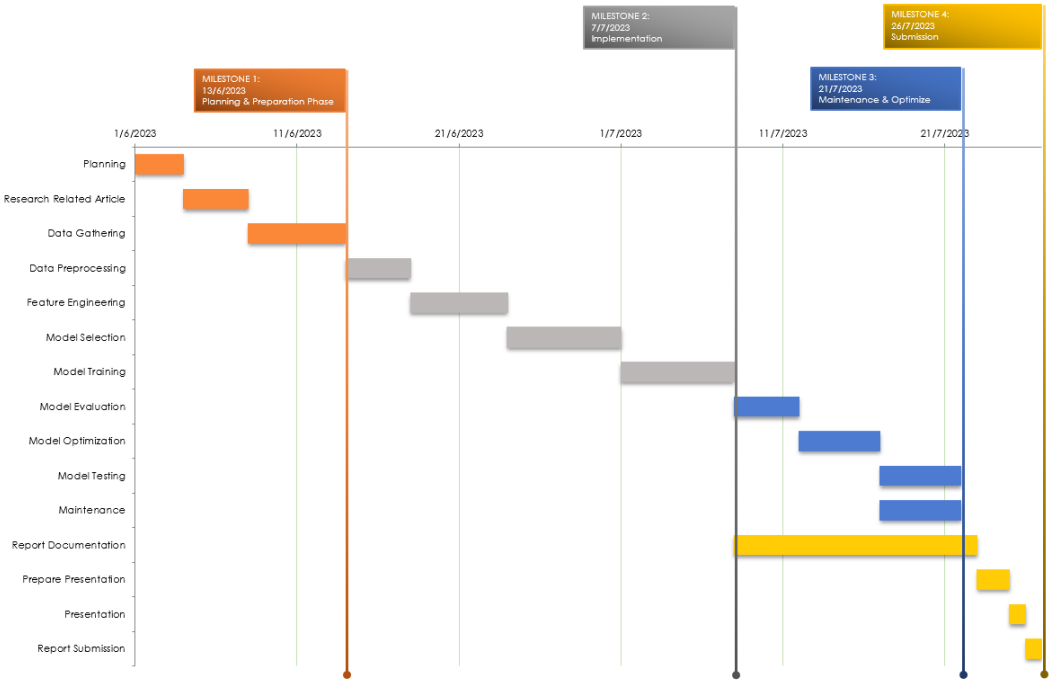
WBS Items	Units/Hrs	Cost/Unit/Hr (RM)	Subtotals (RM)	WBS Level Totals (RM)	Overall
Project Management				**66758**	**26%**
Project Manager	360	50	18000		
Project Team members	1440	30	43200		
Contractors (10% of software development and testing)			5558		
Hardware				**5000**	**2%**
Handheld devices	30	100	3000		
Servers	4	500	2000		
Software				**50800**	**20%**
Licensed software	4	200	800		
Software development			50000		
Testing (10% of total hardware and software costs)			5580	**5580**	**2%**
Training and Support				**87050**	**34%**
Trainee Cost	30	500	15000		
Travel Cost	1	50	50		
Tester Cost	1440	50	72000		
Reserves (20% of total estimate)			43037.6	**43037.6**	**17%**
Total Project Cost estimate				**258225.6**	

Cost Management Chart

The project timeline is divided into four major milestones: Planning & Preparation Phase, Implementation Phase, Maintenance & Optimize Phase, and Submission Phase. The entire project is scheduled to be completed in 80 days.

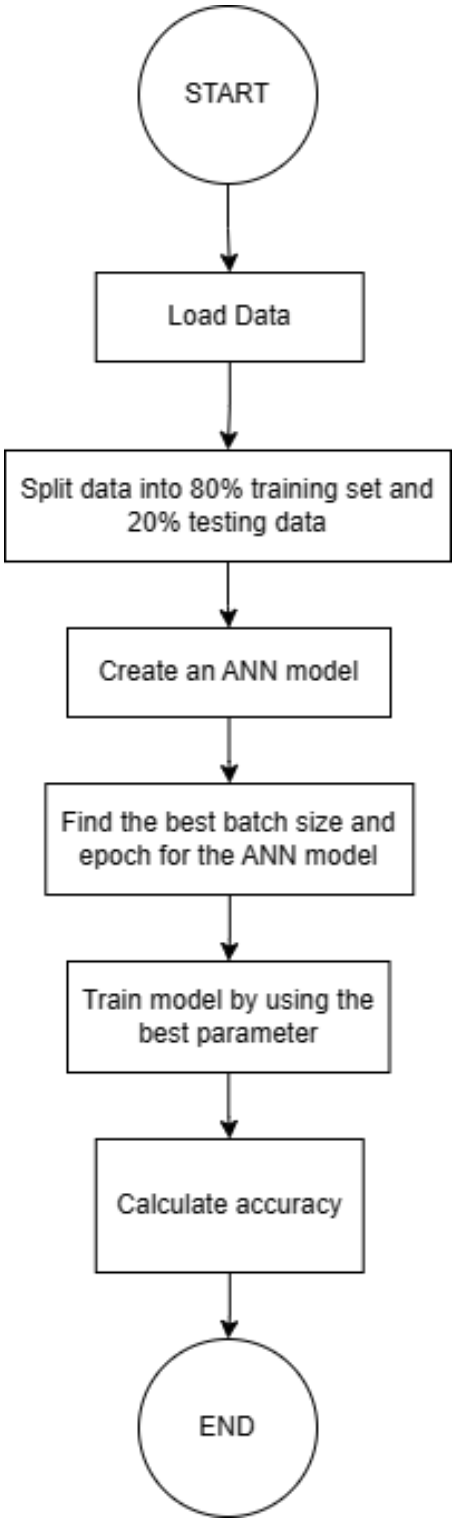
[Predicting Automotive Vehicles Engines]

TASK NAME	START DATE	END DATE	DURATION (days)
Planning	1/6/2023	3/6/2023	3
Research Related Article	4/6/2023	7/6/2023	4
Data Gathering	8/6/2023	13/6/2023	6
Data Preprocessing	14/6/2023	17/6/2023	4
Feature Engineering	18/6/2023	23/6/2023	6
Model Selection	24/6/2023	30/6/2023	7
Model Training	1/7/2023	7/7/2023	7
Model Evaluation	8/7/2023	11/7/2023	4
Model Optimization	12/7/2023	16/7/2023	5
Model Testing	17/7/2023	21/7/2023	5
Maintenance	17/7/2023	21/7/2023	5
Report Documentation	8/7/2023	22/7/2023	15
Prepare Presentation	23/7/2023	24/7/2023	2
Presentation	25/7/2023	25/7/2023	1
Report Submission	26/7/2023	26/7/2023	1



Time Management Chart

FLOW DIAGRAM



Flow diagram of the model development

ALGORITHM AND PROBLEM SOLVING

1. Load the data into workspace.

   Import the dataset to train the model. The dataset used for the project is the Automotive Vehicles Engine Health Dataset in .csv format.

2. Split the data into training set and testing set.

   The target variable and predictors are defined. The target variable is Engine Condition and the predictors include Engine RPM, Lubricating Oil Pressure, Fuel Pressure, Coolant Pressure, Lubricating Oil Temperature, and Coolant Temperature. The predictors are standardised for the prediction. The dataset is then divided into a training set and a testing set, with a ratio of 8:2.

3. Create an ANN model.

   Define the parameter of the model:

   - Input layer: set the input dimension to 6, matching the number of features in the data. The layer has 7 nodes, representing the 6 features and an additional 1 node for bias. The activation function used is the ReLU function. This function will change all negative values to 0 and remain the same values as the output if the value is positive value.
   - Hidden layer: consists of one layer with 6 nodes. The number of nodes is determined by taking the mean of the nodes in the input and output layers. The activation function in the hidden layer is also the ReLU function.
   - For the output layer, include a single node since there is only one target variable. The activation function for this layer is the Sigmoid function, transforming values to a range between 0 and 1.

Create a classifier with the defined parameter. Compile the classifier with these parameters, optimizer = 'adam', loss = 'binary_crossentropy' and metrics = ['accuracy']. After that, fit the neural network on the training data with batch size = 10 and epochs = 10.

4. Find the best batch size and epoch for the ANN model.

   To obtain the highest accuracy, Manual Grid Search is used to identify the optimal parameters for the model. The batch size and epoch are defined which are [5, 10, 15, 20] and [5, 10, 50, 100] respectively to find out which combination provides the highest accuracy.

5. Train the model by using best parameter.

   Use the parameter obtained in the previous step to train the model with the training set.

6. Calculate the model accuracy on test set.

   Make predictions on the testing data using the trained model. Define the probability threshold value to 0.5 to classify whether the engine needs maintenance or not. Next, generate new predictions based on this probability threshold. Finally, calculate precision, recall, f1-score and accuracy from the obtained results.

PROJECT IMPLEMENTATION

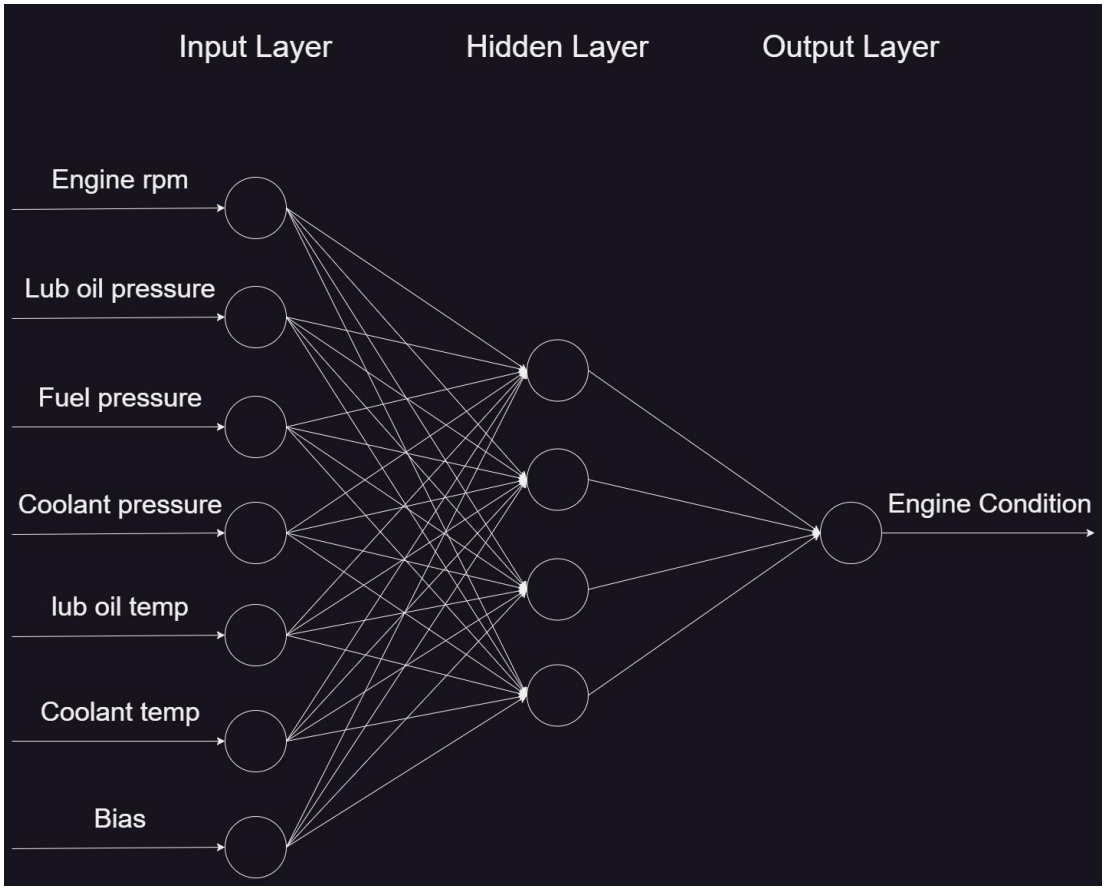


Image of the ANN model

Step 1: Import the required libraries.

```
import pandas as pd
import numpy as np
from matplotlib import pyplot as plt
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from keras.models import Sequential
from keras.layers import Dense
from sklearn import metrics
```

Step 2: Initialise the variables and import the dataset.

```
bbatch = 0
bepoch = 0
baccuracy = 0
minaccuracy = 1

# To remove the scientific notation from numpy arrays
np.set_printoptions(suppress=True)

df = pd.read_csv('C:/Uni/UTeM/SEM 4/NN Neural Network/Project/engine_data.csv')
```

Step 3: Separate the target variable and predictor variables and define it as x and y.

```
# Separate Target Variable and Predictor Variables
TargetVariable = ['Engine_Condition']
Predictors = ['Engine_rpm', 'Lub_oil_pressure', 'Fuel_pressure',
              'Coolant_pressure', 'lub_oil_temp', 'Coolant_temp']

X = df[Predictors].values
y = df[TargetVariable].values
```

Step 4: Standardise the data.

```
# Standardization of data
PredictorScaler = StandardScaler()

# Storing the fit object for later reference
PredictorScalerFit = PredictorScaler.fit(X)

# Generating the standardized values of X and y
X = PredictorScalerFit.transform(X)
```

Step 5: Split the dataset into a training and testing set.

```
# Split the data into training and testing set
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

Step 6: Display the shape of the training and testing set.

```
# Quick sanity check with the shapes of Training and Testing datasets
print(X_train.shape)
print(y_train.shape)
print(X_test.shape)
print(y_test.shape)
```

Step 7: Create an ANN model with the parameters.

```
classifier = Sequential()
# Defining the Input layer
classifier.add(Dense(units=7, input_dim=6, kernel_initializer='uniform', activation='relu'))
# Defining the Hidden layer
classifier.add(Dense(units=4, kernel_initializer='uniform', activation='relu'))
# Defining the Output layer
classifier.add(Dense(units=1, kernel_initializer='uniform', activation='sigmoid'))
classifier.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])
# fitting the Neural Network on the training data
survivalANN_Model = classifier.fit(X_train, y_train, batch_size=10, epochs=10, verbose=1)
```

Step 8: Create a function for finding best parameters by using Manual Grid Search.

```
# Defining a function for finding best hyperparameters
def FunctionFindBestParams(X_train, y_train):
```

Step 9: define the parameters to test in the function.

```
# Defining the list of hyper parameters to try
TrialNumber = 0
batch_size_list = [5, 10, 15, 20]
epoch_list = [5, 10, 50, 100]

SearchResultsData = pd.DataFrame(columns=['TrialNumber', 'Parameters', 'Accuracy'])
```

Step 10: Make a loop for making different parameter combinations and find the model accuracy with the parameters.

```
for batch_size_trial in batch_size_list:
    for epochs_trial in epoch_list:
        TrialNumber += 1

        # Creating the classifier ANN model
        classifier = Sequential()
        classifier.add(Dense(units=7, input_dim=6, kernel_initializer='uniform', activation='relu'))
        classifier.add(Dense(units=4, kernel_initializer='uniform', activation='relu'))
        classifier.add(Dense(units=1, kernel_initializer='uniform', activation='sigmoid'))
        classifier.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])

        survivalANN_Model = classifier.fit(X_train, y_train, batch_size=batch_size_trial, epochs=epochs_trial,
                                           verbose=0)
        # Fetching the accuracy of the training
        Accuracy = survivalANN_Model.history['accuracy'][-1]

        # printing the results of the current iteration
        print(TrialNumber, 'Parameters:', 'batch_size:', batch_size_trial, '-', 'epochs:', epochs_trial,
              'Accuracy:', Accuracy)
```

Step 11: Store the parameters and accuracy into variables.

```
        global bbatch
        global bepoch
        global baccuracy
        global minaccuracy

        if float(Accuracy) < float(minaccuracy):
            minaccuracy = Accuracy

        if float(Accuracy) > float(baccuracy):
            bbatch = batch_size_trial
            bepoch = epochs_trial
            baccuracy = Accuracy

        SearchResultsData = pd.concat([SearchResultsData, pd.DataFrame(data=[[TrialNumber, 'batch_size' +
                                                                              str(batch_size_trial) + '-' + 'epoch' +
                                                                              str(epochs_trial), Accuracy]],
                                                                       columns=['TrialNumber', 'Parameters',
                                                                                'Accuracy'])])
return (SearchResultsData)
```

Step 12: Call the function to find the best parameters.

```
# Calling the function
ResultsData = FunctionFindBestParams(X_train, y_train)
```

Step 13: Display the parameters with its accuracy and visualise it.

```
# Printing the best parameter
print(ResultsData.sort_values(by='Accuracy', ascending=False).head(1))
# Visualizing the results
plt.figure(figsize=(12, 5))
plt.plot(ResultsData.Parameters, ResultsData.Accuracy, label='Accuracy')
plt.xticks(rotation=20)
plt.ylim([round(minaccuracy*0.8, 2), round(baccuracy*1.2, 2)])
plt.show()
```

Step 14: Train the model with the best parameters.

```
# Training the model with best hyperparameters
classifier.fit(X_train, y_train, batch_size=bbatch, epochs=bepoch, verbose=1)
```

Step 15: Make prediction on the testing set by using the model.

```
# Predictions on testing data
Predictions = classifier.predict(X_test)
# Scaling the test data back to original scale
Test_Data = PredictorScalerFit.inverse_transform(X_test)
# Generating a data frame for analyzing the test data
TestingData = pd.DataFrame(data=Test_Data, columns=Predictors)
TestingData['Condition'] = y_test
TestingData['PredictedProb'] = Predictions
```

Step 16: Define the probability threshold value.

```
# Defining the probability threshold
def probThreshold(inpProb):
    if inpProb > 0.5:
        return (1)
    else:
        return (0)
```

Step 17: Reassign the value into output after applying the probability threshold.

```
# Generating predictions on the testing data by applying probability threshold
TestingData['Predicted'] = TestingData['PredictedProb'].apply(probThreshold)
```

Step 18: Display the results and the confusion matrix.

```
print(TestingData.head())
print('\nTesting Accuracy Results: ')
print(metrics.classification_report(TestingData['Condition'], TestingData['Predicted']))
print(metrics.confusion_matrix(TestingData['Condition'], TestingData['Predicted']))
```

OUTPUT

The probability threshold value is set as 0.5, hence the predicted value will be 1 if the predicted probability value is higher than or equal to 0.5, otherwise will be 0. Here are some samples from the table of prediction.

	Engine_rpm	Lub_oil_pressure	...	PredictedProb	Predicted
0	682.0	2.391656	...	0.612737	1
1	605.0	5.466877	...	0.857289	1
2	658.0	3.434232	...	0.589744	1
3	749.0	2.094656	...	0.521460	1
4	676.0	3.538228	...	0.726733	1

[5 rows x 9 columns]

Testing Accuracy Results:

	precision	recall	f1-score	support
0	0.55	0.39	0.46	1459
1	0.69	0.81	0.75	2448
accuracy			0.65	3907
macro avg	0.62	0.60	0.60	3907
weighted avg	0.64	0.65	0.64	3907

Precision is the value of the total number of correct positive predictions made and recall is the value of the total number of positive cases that are correctly predicted. From the testing accuracy results, precision and recall of 1 are higher than 0 which are 0.69 and 0.81 respectively. In short, the accuracy is 0.65 with 3907 of samples.

```
[[ 574  885]
 [ 469 1979]]
```

This is the confusion matrix for the results. We can see that true positive is 574, false positive is 885, false negative is 469 and true negative is 1979 from 3907 sample data.

CONCLUSION

Automatic vehicle engines can be used in many fields to provide power to the connected machine. However, the condition of the engine is hard to predict if the person is not having the skills and knowledge about the engine. This model helps them to predict the condition of the engine from some measurable variables. The model uses Manual Grid Search to find out the best parameters for the prediction to improve the accuracy. This is important because incorrect prediction may cause permanent damage to the engine due to late maintenance. After that, an artificial neural network (ANN) is used to predict the condition of the engine. From the results of the prediction, the accuracy of the prediction is about 65%. This accuracy is acceptable but still has a lot of improvement space. However, the recall of the 1 is 81%, which means most of the engines that need maintenance are predicted correctly. Although the accuracy of the model is not high enough, the model still can ensure most of the engine that is having problems from getting timely maintenance.

REFERENCES

1. Modi, P. (2023, April 5). Automotive Vehicles Engine Health Dataset. Kaggle. https://www.kaggle.com/datasets/parvmodi/automotive-vehicles-engine-health-dataset

2. Vehicle Technologies Office. (2013, November 22). Internal Combustion Engine Basics. Energy.gov. https://www.energy.gov/eere/vehicles/articles/internal-combustion-engine-basics#:~:text=The%20engine%20consists%20of%20a,motion%20drives%20the%20vehicle's%20wheels

3. Rymax Lubricants. (2020, September 9). What is high oil pressure and how to solve it?. Rymax Lubricants. https://www.rymax-lubricants.com/updates/what-is-high-oil-pressure-and-how-to-solve-it/

4. Farukh Hashmi. (2021, November 7). How to use Artificial Neural Networks for classification in python?. Thinking Neurons. https://thinkingneuron.com/how-to-use-artificial-neural-networks-for-classification-in-python/

5. David Landup. (2023, May 17) How to Set Axis Range (xlim, ylim) in Matplotlib. Stack Abuse. https://stackabuse.com/how-to-set-axis-range-xlim-ylim-in-matplotlib/

6. Rob Hyndman. (2010, July 20) How to choose the number of hidden layers and nodes in a feedforward neural network?. Stack Exchange. https://stats.stackexchange.com/questions/181/how-to-choose-the-number-of-hidden-layers-and-nodes-in-a-feedforward-neural-netw

7. Teemu Kanstrén. (2020, September 20). A Look at Precision, Recall, and F1-Score. Towards Data Science. https://towardsdatascience.com/a-look-at-precision-recall-and-f1-score-36b5fd0dd3ec

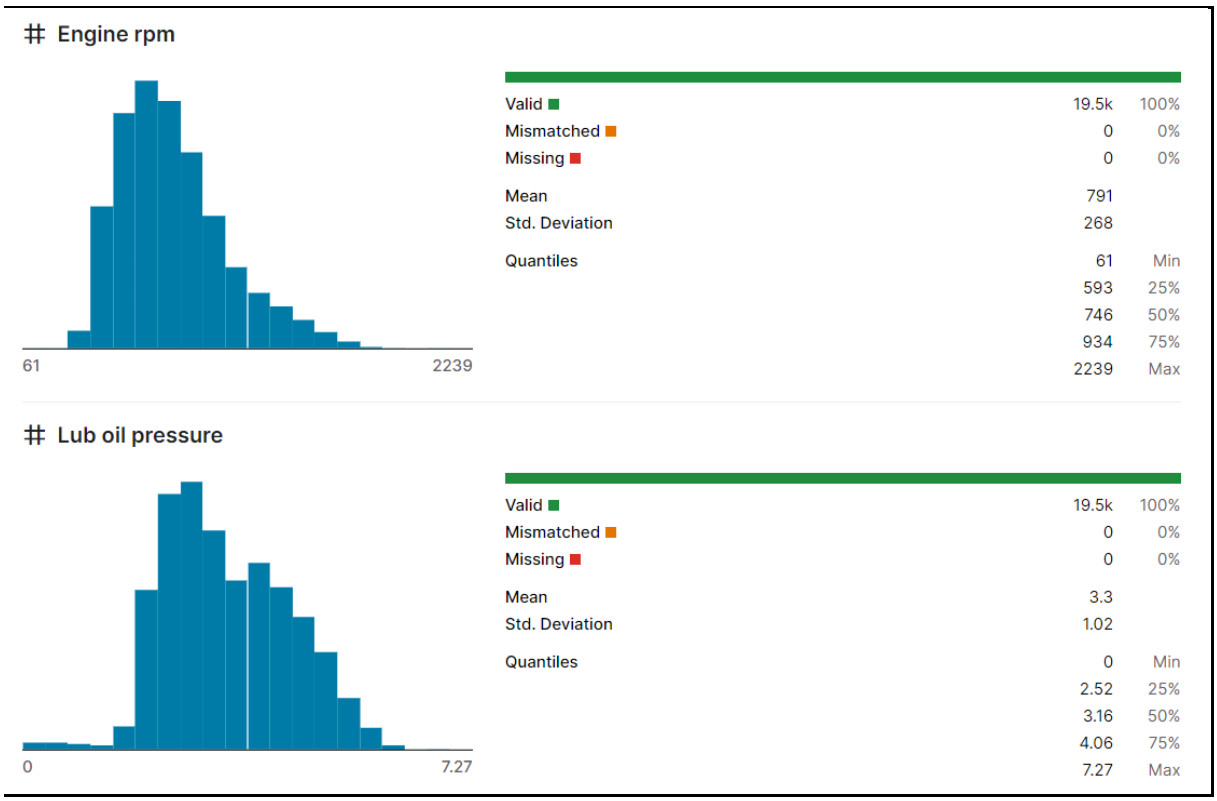
8. Guofeng Ma, Ying Liu, and Shanshan Shang. (2019 September 11). A Building Information Model (BIM) and Artificial Neural Network (ANN) Based System for Personal Thermal Comfort Evaluation and Energy Efficient Design of Interior Space. MDPI. https://www.mdpi.com/2071-1050/11/18/4972

APPENDICES

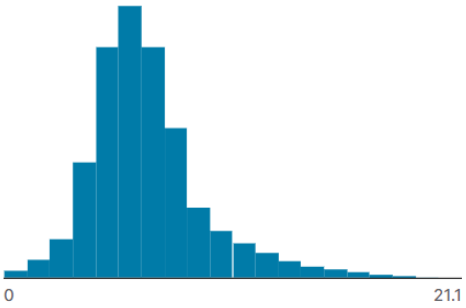
User Manual

1. Open “automotive-vehicles-engine-health-prediction” in github.
2. Download the dataset “engine_data.csv” and “project.py” into computer.
3. Open “project.py” using PyCharm Community Edition 2022.2.3 or above.
4. Import the dataset into the code by changing the path of the dataset in the program.
5. Run the program to get the results.
6. While the program is running, line graph for the model accuracy will pop out, closing it to continue the run.
7. The output in the terminal at bottom is the results of the model.
8. If you want to change the ratio of splitting training and testing data, go to Line 38 and change the test_size value to any number from 0 to 1.
9. If you want to change the parameters, go to Line 61 to change the batch_size_list value with any number greater than 0 to change the size of the data batch, and Line 62 to change the epoch_list value to with any number greater than 0 to change the number of iteration.

Dataset

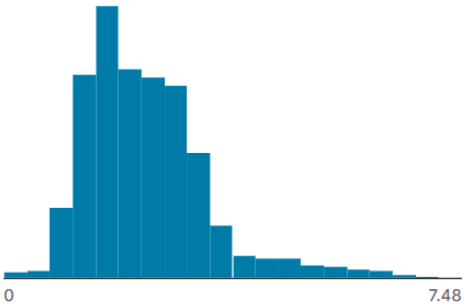


Fuel pressure



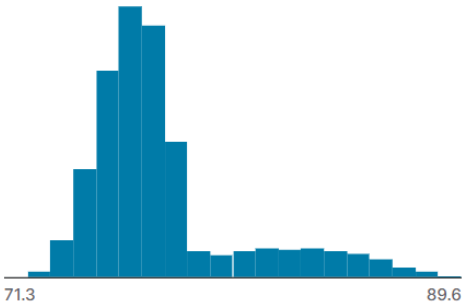
Valid	19.5k	100%
Mismatched	0	0%
Missing	0	0%
Mean	6.66	
Std. Deviation	2.76	
Quantiles	0	Min
	4.92	25%
	6.2	50%
	7.74	75%
	21.1	Max

Coolant pressure



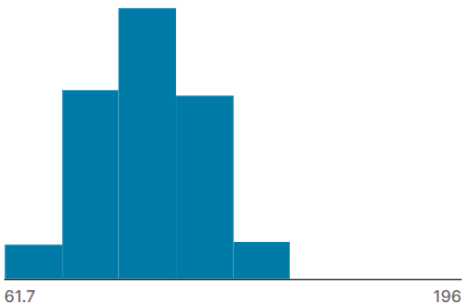
Valid	19.5k	100%
Mismatched	0	0%
Missing	0	0%
Mean	2.34	
Std. Deviation	1.04	
Quantiles	0	Min
	1.6	25%
	2.17	50%
	2.85	75%
	7.48	Max

lub oil temp

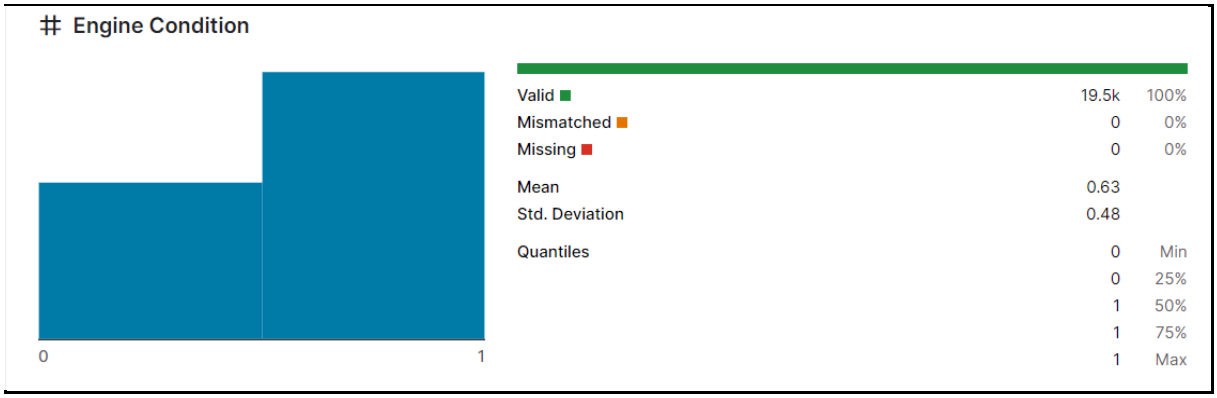


Valid	19.5k	100%
Mismatched	0	0%
Missing	0	0%
Mean	77.6	
Std. Deviation	3.11	
Quantiles	71.3	Min
	75.7	25%
	76.8	50%
	78.1	75%
	89.6	Max

Coolant temp



Valid	19.5k	100%
Mismatched	0	0%
Missing	0	0%
Mean	78.4	
Std. Deviation	6.21	
Quantiles	61.7	Min
	73.9	25%
	78.3	50%
	82.9	75%
	196	Max


Engine Condition
0
1
Valid 19.5k 100%
Mismatched 0 0%
Missing 0 0%
Mean 0.63
Std. Deviation 0.48
Quantiles
0 Min
0 25%
1 50%
1 75%
1 Max