Assignment Cover Sheet

Subject Code: CSCI323

Subject Name: Modern AI

Submission Type: Zip Folder

Assignment Title: CSCI323 Project

Student Name: Mahek Sajid, Samir Fazil, Yash Vadapalli, Sarah Amina Selama

Student Number: 7068293, 7068943, 6507591, 7104017

Student Phone/Mobile No.

Student E-mail: mshs511, sf880, ssyv585, sas966

Lecturer Name: Dr. Patrick Mukala

Due Date: March 11th 2024

Date Submitted: March 11th 2024

PLAGIARISM:	**DECLARATION:**
The penalty for deliberate plagiarism is FAILURE in the subject. Plagiarism is cheating by using the written ideas or submitted work of someone else. UOWD has a strong policy against plagiarism. The University of Wollongong in Dubai also endorses a policy of non-discriminatory language practice and presentation. **PLEASE NOTE**: STUDENTS MUST RETAIN A COPY OF ANY WORK SUBMITTED	I/We certify that this is entirely my/our own work, except where I/we have given fully-documented references to the work of others, and that the material contained in this document has not previously been submitted for assessment in any formal course of study. I/we understand the definition and consequences of plagiarism. **Signature of Student:**

Optional Marks:

Comments:

Lecturer Assignment Receipt (To be filled in by student and retained by Lecturer upon return of assignment)

Subject: Modern AI **Assignment Title: CSCI323 Project**

Student Name: Mehak Sajid, Samir Fazil, Yash Vadapalli, Sarah Amina Selama **Student Number: 7068293, 7068943, 6507591, 7104017**

Due Date: March 11th 2024 **Date Submitted: March 11th 2024**

Signature of Student:

Student Assignment Receipt (To be filled in and retained by Student upon submission of assignment)

Subject: Modern AI **Assignment Title: CSCI323 Project**

Student Name: Mehak Sajid, Samir Fazil, Yash Vadapalli, Sarah Amina Selama **Student Number: 7068293, 7068943, 6507591, 7104017**

Due Date: March 11th 2024 **Date Submitted: March 11th 2024**

Signature of Lecturer

CSCI323 Project

By : Mahek Sajid -7068293, Samir Fazil -7068943, Yash Vadapalli - 6507591, Sarah Selama - 7104017

Table of contents

Table of contents 3
Executive summary 4
Motivation 5
Technical Requirements 6
Main Constructs 7
Implementation 10
Classification Problems 10
Decision Tree Classifiers 10
Random Forest Classifiers 11
Clustering Problems 13
Simple Genetic Algorithm 15
Limitations/Difficulties 24
Decision Tree Model Limitations: 24
Random Forest Model Limitations: 24
K-Means Limitations: 24
Simple Genetic Algorithm Limitations: 23
Conclusions 25

Executive summary

The project aimed to find a real world problem and to use AI to develop a solution. We have found a Dubai-based company called Pan African that works in the construction and infrastructure industry and their goal is to increase their efficiency, productivity and adaptability because the industry struggles with external factors such as delays and weather conditions, time constraints and human error.

Our primary goal was to use AI technology to gain deeper insights that can be helpful for the company and that information can be used in their decision making process. Firstly, we gathered the list of problems that the company was facing, collected datasets that are relevant for developing a solution, and lastly, we had to carefully choose which features are relevant to this problem and how they can be scaled or transformed and which model was well suited for these features and could provide the most accurate results via iterative training and testing and comparing the results of every model.

This report provides insight on how integrating AI technology can be beneficial for companies that still operate traditionally. By highlighting the specific problems the company faces, key findings, implementation and results, we are capable to draw certain conclusions because of the ability of AI models to provide highly accurate results and predictions.

Despite its benefits, there are still a few drawbacks to this technology, such as requiring continuous training and testing until you gain accurate results, which can be time-consuming. Moreover, this industry relies on human labour, and humans do not operate in the automated manner that the AI does, which could render certain predictions as useless if the human operates differently.

Nonetheless, we believe that integrating AI solutions could prove to be a useful tool for a company working traditionally.

Motivation

Our company identified three problems the client struggled with: route optimization, grouping of trucks and package to truck assignment. All three problems stem from scheduling, which is a crucial aspect for a company working in the construction industry. Delays and lack of efficiency are detrimental to the company because they can cause supply chain disruption, which leads to a domino effect that will impact the company directly and their business associates.
Previous approaches to solving these problems involved manual data analysis and scheduling, and despite using software applications that stored that data, these approaches were not the most ideal as they are time-consuming, prone to human error, and lacked the ability to use the data to its full potential.

We consider AI to be a powerful solution for the problems because of its ability to process large amounts of data quickly, apply machine learning algorithms, and adapt to changes due to its automatic nature. This allows us to uncover connections within the various factors involved in the job. These connections can form a specific pattern that correlates to a specific impact, which in turn provides deeper insight for the analysts to improve their decision-making process, which in turn improves efficiency and the performance of the company.

Technical Requirements

This project was implemented using the Python programming language and a number of libraries, including NumPy, Pandas, and Matplotlib. High-level, interpreted Python is a simple language to learn and use. It features a sizable standard library and a sizable community that offers many third-party packages to carry out different tasks.

We used Google Colab as our development environment for this project. Google Colab is an online tool that enables users to create and execute Python code in a Jupyter notebook setting. It is a smart choice for carrying out resource-intensive tasks because it offers free access to computational resources like GPUs and TPUs.

We used a variety of libraries, including NumPy, Pandas, and Matplotlib, in addition to these tools to do data analysis, visualization, and modification. A library called NumPy offers assistance for numerical operations on arrays and matrices. High-performance, simple-to-use data structures and tools for data analysis are offered by the Pandas package. A library called Matplotlib offers numerous plot and visualization genres.

Overall, the use of Python, Google Colab, and several libraries allowed for the creation of a reliable and effective solution to the problem at hand.

Main Constructs

We have used 4 Machine Learning models for this project which are Decision Tree classifiers and Random Forest Classifiers for the classification problems, K means Algorithm for the Clustering problems and Simple Genetic Algorithm for the Assignment Problems.

Decision Tree Classifiers -
Our code is a roadmap for training and evaluating a classification model using scikit-learn. It starts by loading a dataset and dividing it into training and testing parts. The features are then preprocessed—numeric ones get scaled, and categorical ones are transformed. The heart of this process is a DecisionTreeClassifier model, all neatly organized in a pipeline. This structured approach simplifies tasks from training the model to predicting outcomes on the test set, with performance evaluation through accuracy metrics and a classification report. The use of pipelines and transformers enhances code clarity and maintainability.

Random Forest Classifiers -
For this code we had imported essential libraries, and the same dataset was loaded. Feature engineering is introduced by creating a new feature, 'Distance per Fuel Efficiency.' Features and the target variable are then selected for model training, and the data is split into training and testing sets. Preprocessing steps are defined for both numeric and categorical features using pipelines and a ColumnTransformer. The distinctive element is the incorporation of a Random Forest Classifier into the pipeline as the chosen model. The code trains the model, makes predictions on the test set, and evaluates performance through accuracy metrics and a classification report. The code's structure emphasizes the inclusion of feature engineering and the application of a Random Forest model for enhanced classification capabilities.

K-Means Clustering -
We had performed K-Means clustering on a dataset containing geographical and environmental features related to routes. After loading the dataset and selecting relevant features, numerical and categorical transformers are created for preprocessing. The code utilizes a ColumnTransformer to apply appropriate transformations to each feature. The chosen number of clusters (K) is set to

3, and a pipeline is constructed, combining preprocessing with K-Means clustering. The pipeline is then applied to fit and predict clusters, with the results visualized on a scatter plot. Additionally, the code prints the cluster assignments for a subset of routes and allows for further analysis and evaluation of the clusters based on specific objectives.

Simple Genetic Algorithm -
Our code implements a Simple Genetic Algorithm (SGA) for optimizing truck assignments to packages based on their weights and distances. The algorithm begins by randomly assigning trucks to packages and then iteratively refines the assignments through genetic operations such as selection, crossover, and mutation. The SGA aims to minimize the maximum load on any truck, providing an optimized solution.

The code defines functions for random assignment, cost calculation, roulette wheel selection, crossover, mutation, and the main SGA process. Truck and package data are read from our dataset, and SGA parameters such as population size, mutation rate, and crossover rate are set. The algorithm is executed, and the final results, including both a normally optimized solution and the best solution found, are printed, displaying the assigned packages to trucks along with their respective costs. The algorithm concludes with a local search to further refine the best solution.

These were the datasets that we worked with.
Dataset 1- Has information about all trucks and other vehicles and their details.

ID	Equipment	Task ID	Load Capa	Fuel Efficie	Operation	Route Star	Route Star	Route End	Route End	Distance (k	Terrain Ty	Weather C	Scheduled	Actual Sta	Completio	Status
EH3262-N	Tyre Handl	770-78	55	5.93	12:13	11.16496	124.3923	12.69206	109.2222	27.38	Gravel	Rainy	10:30	2:21	19:14	Delayed
XO0539-RI	Tyre Handl	202-90	95	4.62	7:01	-29.7725	31.05754	60.811	21.44573	11.38	Sandy	Windy	21:55	10:13	18:33	Delayed
NV8033-E\	Articulatec	952-02	200	1.02	16:20	-31.6577	-60.783	41.21154	-8.17	47.29	Arid/Deser	Sunny	7:57	1:57	9:20	Completed
ZU7922-RI	Mining die	381-14	145	2.48	22:36	45.82664	20.59376	56.3997	61.87998	48.9	Mountainc	Windy	2:11	9:55	5:15	Delayed
GE8785-SC	Articulatec	250-54	200	3.36	8:18	28.87157	105.4417	-7.07952	109.0869	24.48	Clay	Rainy	17:17	8:53	17:24	Delayed
WT6802-V	Water truc	198-14	130	1.21	23:36	55.67747	12.58342	40.38293	-8.72415	49.59	Sandy	Rainy	2:30	11:48	3:18	Completed
GV2229-K/	Mining die	503-21	280	3.04	3:26	19.44584	-99.1498	29.84411	115.5612	38.86	Rocky	Snowy	14:30	15:24	13:43	Delayed
EV8358-AI	Service tru	587-76	65	2.36	10:19	16.28919	119.9056	11.06788	-63.9202	43.56	Arid/Deser	Cloudy	23:55	3:13	7:20	Completed
TJ3029-AF	Service tru	433-33	285	1.19	1:37	-6.50049	106.656	59.169	25.20835	21.37	Rocky	Sunny	17:38	15:55	22:46	Completed
IN3535-YC	Wheel Loa	969-51	240	4.27	8:06	11.86483	121.878	35.92277	139.7492	32.51	Wetland/N	Snowy	21:00	16:39	6:18	Delayed
QK1086-N	Wheel Loa	777-72	290	4.79	13:31	34.67189	133.8965	39.78695	-8.82456	29.41	Savannah	Windy	19:54	9:21	21:05	Completed
AE0336-FT	Articulatec	257-14	215	4.79	2:45	29.40993	120.1289	10.32918	123.8864	10.31	Rocky	Sunny	12:27	5:47	8:33	Delayed
YA1618-SX	Tyre Handl	072-36	90	5.65	2:58	-9.57158	-35.8254	53.91841	19.74428	48.95	Loamy	Snowy	4:00	10:34	7:41	Completed
CC2669-KI	Excavators	690-86	275	4.92	1:42	-21.9703	-44.92	59.85299	30.24598	7.97	Arid/Deser	Snowy	20:30	0:11	2:37	Completed
SE0953-CE	Track Doze	623-62	250	5.72	1:30	-8.21742	113.5517	49.72661	19.8365	18.89	Wetland/N	Cloudy	18:19	23:46	10:25	Delayed
OU2562-U	Excavators	410-71	80	2.56	13:54	55.86328	37.6217	33.76846	-118.202	36.94	Gravel	Rainy	14:37	3:42	8:07	Completed
PX8029-W	Articulatec	437-88	100	5.62	20:36	31.42924	31.12521	49.07011	16.46492	33.32	Rocky	Sunny	20:13	3:58	7:42	Delayed
UM3262-I	Water truc	647-93	85	2.35	1:04	15.55273	48.51639	36.40459	139.2613	5.2	Savannah	Cloudy	0:48	20:51	23:41	Completed
UK9066-B/	Excavators	045-92	150	5.55	15:38	20.536	-103.433	15.48446	-84.3568	6.63	Clay	Rainy	3:45	10:42	21:13	Delayed
CG0528-N	Track Doze	041-62	105	3.91	12:49	50.83105	34.95284	34.6911	106.2211	7.13	Gravel	Sunny	2:28	20:57	0:39	Completed
YU3620-TJ	Water truc	000-69	165	5.01	10:55	48.64055	2.326932	-7.8817	113.7782	41.95	Loamy	Snowy	18:04	17:51	16:17	Completed
HV7317-BI	Excavators	816-58	160	2.11	19:40	6.675981	100.6647	-9.7811	34.52267	39.21	Sandy	Cloudy	17:45	9:41	11:26	Delayed
OT7366-R.	Tyre Handl	015-97	270	3.39	9:28	31.6426	130.5509	41.33364	-8.57494	42.04	Rocky	Rainy	19:43	7:25	15:15	Completed
ZQ3506-G/	Service tru	460-21	210	3.86	12:15	26.87986	113.6462	53.97326	123.9092	24.08	Arid/Deser	Windy	11:47	20:33	18:20	Completed
MK3067-P	Track Doze	888-07	100	2.24	14:08	59.20748	37.11816	24.7168	118.4763	34.81	Savannah	Windy	0:46	11:38	7:30	Delayed
KC4676-D(	Tyre Handl	656-56	225	4.28	2:42	15.79903	120.4542	48.8168	2.640235	47.17	Forested	Windy	8:07	17:30	6:34	Delayed
KV3183-PI	Articulatec	697-70	120	3.58	12:27	11.747	11.96619	46.88431	-96.7848	36.83	Rocky	Windy	11:53	18:24	10:31	Completed
XT9528-V(	Water truc	620-83	110	2.28	5:13	46.29935	4.826579	31.38595	120.9361	35.23	Forested	Snowy	23:41	11:24	22:04	Delayed
KH9563-RI	Wheel Loa	396-45	295	4.71	6:47	-13.1639	-74.2236	10.48532	5.145813	23.86	Arid/Deser	Cloudy	19:47	4:07	17:14	Delayed

Dataset 2- Has information about all trucks and packages.

TruckID	TruckCapa	PackageID	PackageW	PackageDistance(kms)
EH3262-N	700	ABC123	123	100
XO0539-RI	550	XYZ789	456	200
NV8033-E	990	DEF456	789	300
ZU7922-RI	800	GHI789	234	400
GE8785-SC	180	JKL012	567	500
WT6802-V	550	MNO345	890	150
GV2229-KA	1200	PQR678	321	250
EV8358-AI	3000	STU901	654	350
TJ3029-AF	750	VWX234	987	450
IN3535-YC	160	YZA567	543	50
QK1086-N	1100	BCD890	876	175
AE0336-FT	400	EFG123	210	225
YA1618-SX	700	HIJ456	345	275
CC2669-KI	1400	KLM789	678	325
SE0953-CE	3300	NOP012	901	375
OU2562-U	1700	QRS345	432	125
PX8029-W	2000	TUV678	765	275
UM3262-I	1100	WXY901	198	425
UK9066-B	2900	ZAB234	876	475
CG0528-N	2200	CDE567	543	50
YU3620-TJ	2700	FGH890	123	200
HV7317-BI	1600	IJK123	456	300
OT7366-R	3400	LMN456	789	400
ZQ3506-GA	2300	OPQ789	234	100
MK3067-P	1800	RST012	567	250
KC4676-DC	2500	UVW345	890	350
KV3183-PI	1200	XYZ678	321	450
XT9528-VC	2800	123ABC	654	150
KH9563-RI	1500	456DEF	987	475

Implementation

Classification Problems

We were provided with a route assignment problem. The goal is to predict the optimal route (either 'Completed' or 'Delayed') for trucks/excavators based on various features such as load capacity, fuel efficiency, distance, terrain type, and weather conditions.
We had worked with 2 Machine learning models to see which provides us with a better accuracy score. We had used Decision Tree and Random Forest Classifiers.

Decision Tree Classifiers

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, OneHotEncoder
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import classification_report, accuracy_score

df = pd.read_csv('323 Dataset.csv')
features = df[['Load Capacity(in tons)', 'Fuel Efficiency', 'Distance
(km)', 'Terrain Type', 'Weather Condition']]
```

```
target = df['Status']

X_train, X_test, y_train, y_test = train_test_split(features, target,
test_size=0.2, random_state=42)

numeric_features = ['Load Capacity(in tons)', 'Fuel Efficiency', 'Distance
(km)']
numeric_transformer = Pipeline(steps=[('scaler', StandardScaler())])

categorical_features = ['Terrain Type', 'Weather Condition']
categorical_transformer = Pipeline(steps=[('onehot',
OneHotEncoder(handle_unknown='ignore'))])

preprocessor = ColumnTransformer(
    transformers=[
        ('num', numeric_transformer, numeric_features),
        ('cat', categorical_transformer, categorical_features)
    ])

model = Pipeline(steps=[('preprocessor', preprocessor),
                        ('classifier',
DecisionTreeClassifier(random_state=42))])

model.fit(X_train, y_train)
y_pred = model.predict(X_test)
print("Accuracy:", accuracy_score(y_test, y_pred))
print("Classification Report:\n", classification_report(y_test, y_pred))
```

The output -

```
Accuracy: 0.415
Classification Report:
              precision    recall  f1-score   support

   Completed       0.41      0.40      0.41       100
     Delayed       0.42      0.43      0.42       100

    accuracy                           0.41       200
   macro avg       0.41      0.42      0.41       200
weighted avg       0.41      0.41      0.41       200
```

Random Forest Classifiers

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report, accuracy_score
from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import StandardScaler, OneHotEncoder
from sklearn.pipeline import Pipeline


df = pd.read_csv('323 Dataset.csv')
df['Distance per Fuel Efficiency'] = df['Distance (km)'] / df['Fuel
Efficiency']


features = df[['Load Capacity(in tons)', 'Fuel Efficiency', 'Distance
(km)', 'Terrain Type', 'Weather Condition', 'Distance per Fuel
Efficiency']]
target = df['Status']

X_train, X_test, y_train, y_test = train_test_split(features, target,
test_size=0.2, random_state=42)

numeric_features = ['Load Capacity(in tons)', 'Fuel Efficiency', 'Distance
(km)', 'Distance per Fuel Efficiency']
numeric_transformer = Pipeline(steps=[('scaler', StandardScaler())])

categorical_features = ['Terrain Type', 'Weather Condition']
categorical_transformer = Pipeline(steps=[('onehot',
OneHotEncoder(handle_unknown='ignore'))])

preprocessor = ColumnTransformer(
    transformers=[
        ('num', numeric_transformer, numeric_features),
        ('cat', categorical_transformer, categorical_features)
    ])


model = Pipeline(steps=[('preprocessor', preprocessor),
```

```
                        ('classifier',
RandomForestClassifier(random_state=42))])

model.fit(X_train, y_train)
y_pred = model.predict(X_test)
print("Accuracy:", accuracy_score(y_test, y_pred))
print("Classification Report:\n", classification_report(y_test, y_pred))
```

Output -

```
Accuracy: 0.54
Classification Report:
              precision    recall  f1-score   support

   Completed       0.54      0.51      0.53       100
     Delayed       0.54      0.57      0.55       100

    accuracy                           0.54       200
   macro avg       0.54      0.54      0.54       200
weighted avg       0.54      0.54      0.54       200
```

When we compare the 2 models we can see that The Random Forest model shows a notable improvement in overall accuracy compared to the Decision Tree model (from 41.5% to 54%) and The Random Forest model demonstrates more balanced performance between precision, recall, and F1-Score for both 'Completed' and 'Delayed' classes.

Clustering Problems

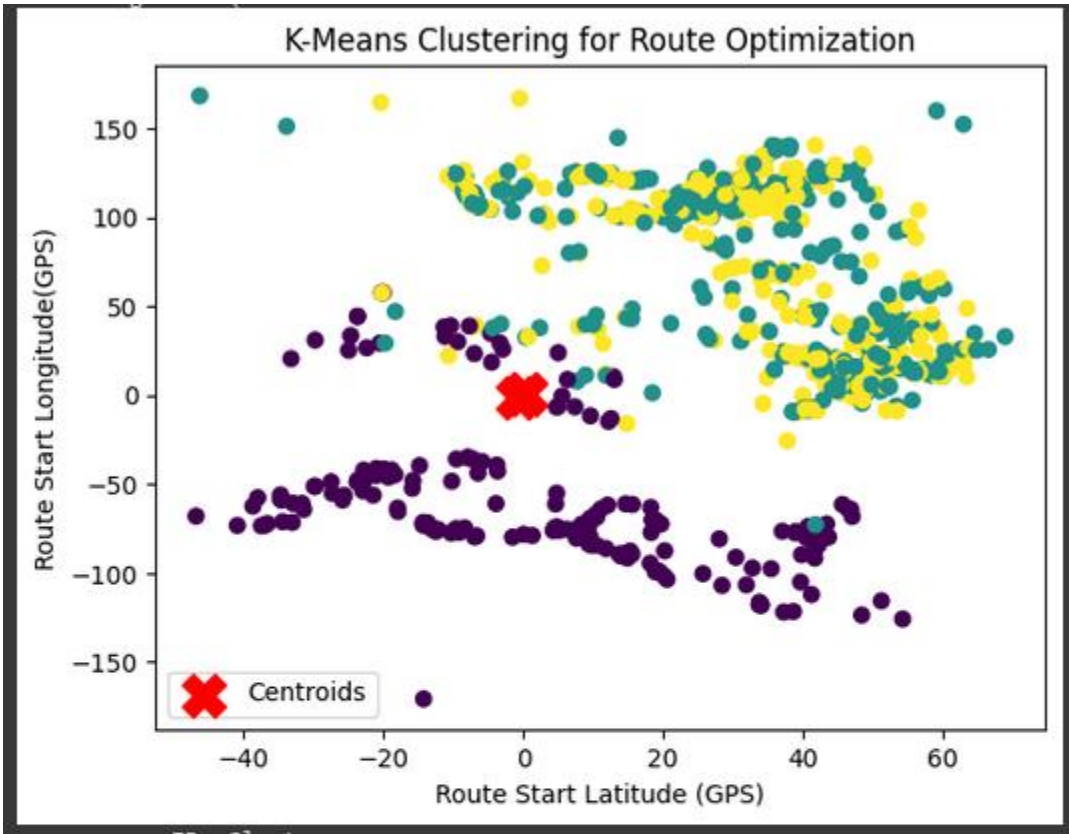
We were given the problem of Route Grouping. The goal here is to group similar routes together to facilitate efficient resource allocation. We'll use the K-Means Clustering algorithm for this task.

```
import pandas as pd
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler, OneHotEncoder
from sklearn.compose import ColumnTransformer
```

```
from sklearn.pipeline import Pipeline
import matplotlib.pyplot as plt

data = pd.read_csv('323 Dataset.csv')
features = ['Route Start Latitude (GPS)', 'Route Start Longitude(GPS)',
'Distance (km)', 'Terrain Type', 'Weather Condition']

numerical_features = ['Route Start Latitude (GPS)', 'Route Start
Longitude(GPS)', 'Distance (km)']
categorical_features = ['Terrain Type', 'Weather Condition']

numerical_transformer = StandardScaler()
categorical_transformer = Pipeline(steps=[
    ('onehot', OneHotEncoder(handle_unknown='ignore'))
])


preprocessor = ColumnTransformer(
    transformers=[
        ('num', numerical_transformer, numerical_features),
        ('cat', categorical_transformer, categorical_features)
    ])


chosen_k = 3

pipeline = Pipeline(steps=[
    ('preprocessor', preprocessor),
    ('kmeans', KMeans(n_clusters=chosen_k, random_state=42))
])


data['Cluster'] = pipeline.fit_predict(data[features])


plt.scatter(data['Route Start Latitude (GPS)'], data['Route Start
Longitude(GPS)'], c=data['Cluster'], cmap='viridis')
plt.scatter(pipeline.named_steps['kmeans'].cluster_centers_[:, 0],
pipeline.named_steps['kmeans'].cluster_centers_[:, 1], s=300, marker='X',
c='red', label='Centroids')
```

```
plt.title('K-Means Clustering for Route Optimization')
plt.xlabel('Route Start Latitude (GPS)')
plt.ylabel('Route Start Longitude(GPS)')
plt.legend()
plt.show()


print(data[['ID', 'Cluster']].head(100))
```

Output



	ID	Cluster
0	EH3262-NH	1
1	XO0539-RP	0
2	NV8033-EW	0
3	ZU7922-RD	2
4	GE8785-SC	1
..	...	...
95	UQ2038-FJ	0
96	BG3586-QX	1
97	DJ6056-KT	1
98	OV7286-OD	1
99	GF1266-WR	0

[100 rows x 2 columns]

In this code, a K-Means clustering approach is applied to a dataset containing route information for heavy machinery and mining sites. The dataset is preprocessed using a ColumnTransformer, which scales numerical features and one-hot encodes categorical features. The K-Means algorithm with three clusters is then employed to group routes based on their geographical coordinates, distance, terrain type, and weather conditions. The resulting clusters are visualized on a scatter plot, where each point represents a route with colors indicating the assigned cluster. Additionally, red 'X' markers denote the centroids of each cluster. The output includes a DataFrame displaying the first 100 rows of the dataset with the 'ID' and corresponding cluster assignments.The output DataFrame displays the 'ID' of each route alongside its assigned cluster label, providing a clear overview of the clustering results. For example, route 'EH3262-NH' is in Cluster 1, 'XO0539-RP' is in Cluster 0, and 'NV8033-EW' is also in Cluster 0, and so on. This clustering approach aids in optimizing route planning for heavy machinery, facilitating resource allocation and operational efficiency.

Simple Genetic Algorithm

We were provided with an assignment problem. The goal is to minimize the maximum load of any given truck, thus improving the overall efficiency of transportation.

```
import csv
import random

# Function to randomly assign trucks
def random_assignment(trucks, packages):
    assignments = []
    for package in packages:
        truck = random.choice(trucks)
        assignments.append((package, truck))
    return assignments
# Function to calculate the cost of assignments
def calculate_cost(individual, trucks):
    truck_loads = {truck['name']: 0 for truck in trucks}
    for package, truck in individual:
        truck_loads[truck['name']] += package['weight']
    max_load = max(truck_loads.values())
    return max_load

# Function for roulette wheel selection
def roulette_selection(fitness, population_size):
```

```
    total_fitness = sum(fitness)
    selected_value = random.uniform(0, total_fitness)
    cumulative_fitness = 0
    for i, f in enumerate(fitness):
        cumulative_fitness += f
        if cumulative_fitness >= selected_value:
            return i

# Function for crossover
def crossover(parent1, parent2):
    crossover_point = random.randint(1, len(parent1) - 1)
    child1 = parent1[:crossover_point] + parent2[crossover_point:]
    child2 = parent2[:crossover_point] + parent1[crossover_point:]
    return child1, child2

# Function for mutation
def mutation(individual, trucks):
    mutated_individual = []
    for gene in individual:
        if random.random() < MUTATION_RATE:
            mutated_gene = random.choice(trucks)
        else:
            mutated_gene = gene
        mutated_individual.append(mutated_gene)
    return mutated_individual

# Function for Simple Genetic Algorithm (SGA)
def sga(trucks, packages):
    population = [random_assignment(trucks, packages) for _ in
range(POP_SIZE)]

    fitness = [calculate_cost(individual, trucks) for individual in
population]

    best_solution = population[fitness.index(min(fitness))]
    best_fitness = min(fitness)


    while True:
```

```
        selected_indices = [roulette_selection(fitness, POP_SIZE) for _ in
range(POP_SIZE)]
        offspring = []
        for i in range(POP_SIZE):
            parent1 = population[selected_indices[i]]
            if random.random() < CROSSOVER_RATE and i < POP_SIZE - 1:
                parent2 = population[selected_indices[i + 1]]
                child1, child2 = crossover(parent1, parent2)
                offspring.append(child1)
                offspring.append(child2)
            else:
                offspring.append(parent1)

        offspring_fitness = [calculate_cost(ind, trucks) for ind in
offspring]

        for i in range(POP_SIZE):
            if offspring_fitness[i] < fitness[i]:
                population[i] = offspring[i]
                fitness[i] = offspring_fitness[i]

        if abs(best_fitness - min(fitness)) < 0.5:
            break

    # Perform local search
    current_solution = best_solution
    current_fitness = best_fitness
    while True:
        new_fitness = calculate_cost(current_solution, trucks)
        if abs(best_fitness - new_fitness) < 0.5:
            break

        current_fitness = new_fitness


    normal_solution = population[0]
    normal_cost = fitness[0]

    print("\n--- Results ---")
    print("Normal Solution:")
```

```
    for package, truck in normal_solution:
        print(f"  Package: {package['name']}, Truck: {truck['name']}")
    print(f"Normal Cost: {normal_cost:.2f}")
    print("\nBest Solution:")
    for package, truck in best_solution:
        print(f"  Package: {package['name']}, Truck: {truck['name']}")


    print(f"Best Cost: {best_fitness:.2f}")

    return normal_solution, normal_cost, best_solution, best_fitness

random.seed(42)
POP_SIZE = 50
MUTATION_RATE = 0.1
CROSSOVER_RATE = 0.8


trucks = []
packages = []

with open('trucks.csv', 'r') as csvfile:
    reader = csv.DictReader(csvfile)
    for row in reader:
        truck_capacity = row['TruckCapacity(kgs)']
        package_weight = row['PackageWeight(kgs)']
        package_distance = row['PackageDistance(kms)']

        if truck_capacity and package_weight and package_distance:
            truck = {'name': row['TruckID'], 'capacity':
int(truck_capacity)}
            package = {'name': row['PackageID'], 'weight':
int(package_weight), 'distance': int(package_distance)}
            trucks.append(truck)
            packages.append(package)

normal_solution, normal_cost, best_solution, best_fitness = sga(trucks,
packages)
```

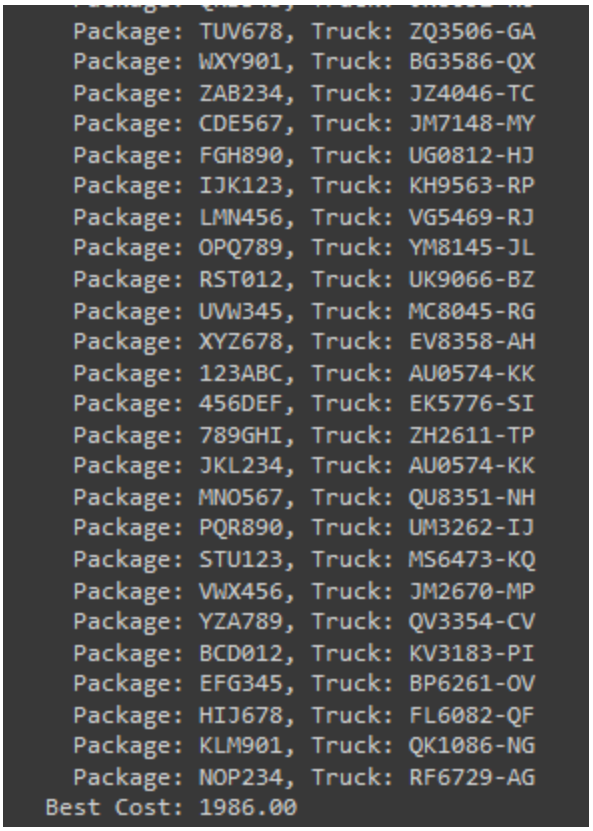
Output -
The Normal Solution

```
--- Results ---
Normal Solution:
  Package: ABC123, Truck: GV2229-KA
  Package: XYZ789, Truck: UG0812-HJ
  Package: DEF456, Truck: ZH2611-TP
  Package: GHI789, Truck: IN9539-VI
  Package: JKL012, Truck: QL2870-ZN
  Package: MNO345, Truck: XG8954-CF
  Package: PQR678, Truck: TJ3029-AF
  Package: STU901, Truck: SE0953-CE
  Package: VWX234, Truck: YT5059-MV
  Package: YZA567, Truck: XX6481-OE
  Package: BCD890, Truck: JZ4046-TC
  Package: EFG123, Truck: NV8033-EW
  Package: HIJ456, Truck: CZ7109-YC
  Package: KLM789, Truck: GZ8109-YS
  Package: NOP012, Truck: UW8617-XE
  Package: QRS345, Truck: UG0812-HJ
  Package: TUV678, Truck: UU0697-RZ
  Package: WXY901, Truck: UK9066-BZ
  Package: ZAB234, Truck: EW9020-EY
  Package: CDE567, Truck: TX3796-ID
  Package: FGH890, Truck: EH3262-NH
  Package: IJK123, Truck: ET0168-OD
  Package: LMN456, Truck: VH2985-GB
  Package: OPQ789, Truck: UG0812-HJ
  Package: RST012, Truck: UW8617-XE
  Package: UVW345, Truck: ZH2611-TP
  Package: XYZ678, Truck: ZQ3506-GA
  Package: 123ABC, Truck: IL0285-EL
  Package: 456DEF, Truck: IL0285-EL
  Package: 789GHI, Truck: QK1086-NG
  Package: UVW901, Truck: QU8351-NH
  Package: XYZ234, Truck: KQ2596-NX
  Package: 789ABC, Truck: TJ3029-AF
  Package: DEF567, Truck: QU8351-NH
  Package: GHI890, Truck: TH5775-AL
  Package: JKL234, Truck: OA5609-FK
  Package: MNO567, Truck: OD9287-QS
  Package: PQR890, Truck: OA5609-FK
  Package: STU123, Truck: CA9494-DB
  Package: VWX456, Truck: NV8033-EW
  Package: YZA789, Truck: MS6473-KQ
  Package: BCD012, Truck: JM7148-MY
  Package: EFG345, Truck: YK5247-JX
  Package: HIJ678, Truck: HV7317-BF
  Package: KLM901, Truck: TY7440-AA
  Package: NOP234, Truck: YK7336-LP
  Package: QRS567, Truck: YU3620-TJ
  Package: TUV890, Truck: GV2229-KA
```

```
    Package: GHI890, Truck: YA1618-SX
    Package: JKL012, Truck: GV2229-KA
    Package: MNO345, Truck: RV3412-NF
    Package: PQR678, Truck: UW8617-XE
    Package: STU901, Truck: CA9494-DB
    Package: VWX234, Truck: GU7620-PZ
    Package: YZA567, Truck: CG0528-NB
    Package: BCD890, Truck: GV2229-KA
    Package: EFG123, Truck: KQ2596-NX
    Package: HIJ456, Truck: EH3262-NH
    Package: KLM789, Truck: TX3796-ID
    Package: NOP012, Truck: AE0336-FT
    Package: QRS345, Truck: EW9020-EY
    Package: TUV678, Truck: YC9978-HS
    Package: WXY901, Truck: CZ7109-YC
    Package: ZAB234, Truck: XX6481-OE
    Package: CDE567, Truck: GY2228-FN
    Package: FGH890, Truck: KC4676-DQ
    Package: IJK123, Truck: YA1618-SX
    Package: LMN456, Truck: ZU7922-RD
    Package: OPQ789, Truck: KV3183-PI
    Package: RST012, Truck: HV7317-BF
    Package: UVW345, Truck: IN9539-VI
    Package: XYZ678, Truck: EW9020-EY
    Package: 123ABC, Truck: QK1086-NG
    Package: 456DEF, Truck: JM7148-MY
    Package: 789GHI, Truck: SE0953-CE
    Package: JKL234, Truck: DE6591-UL
    Package: MNO567, Truck: OD9287-QS
    Package: PQR890, Truck: RV3412-NF
    Package: STU123, Truck: JM7148-MY
    Package: VWX456, Truck: GU7620-PZ
    Package: YZA789, Truck: YC9978-HS
    Package: BCD012, Truck: DK7193-QG
    Package: EFG345, Truck: AE0336-FT
    Package: HIJ678, Truck: FL6082-QF
    Package: KLM901, Truck: TH5775-AL
    Package: NOP234, Truck: MS6473-KQ
  Normal Cost: 2467.00
```

The Best Solution -

```
Best Solution:
  Package: ABC123, Truck: HG0183-VW
  Package: XYZ789, Truck: DJ6056-KT
  Package: DEF456, Truck: SE0953-CE
  Package: GHI789, Truck: GE8785-SC
  Package: JKL012, Truck: YC9978-HS
  Package: MNO345, Truck: ZH2611-TP
  Package: PQR678, Truck: ET0168-OD
  Package: STU901, Truck: OV7286-OD
  Package: VWX234, Truck: NV8033-EW
  Package: YZA567, Truck: UG0812-HJ
  Package: BCD890, Truck: KV3183-PI
  Package: EFG123, Truck: TJ3029-AF
  Package: HIJ456, Truck: YA1618-SX
  Package: KLM789, Truck: XX6481-OE
  Package: NOP012, Truck: GE8785-SC
  Package: QRS345, Truck: RF6729-AG
  Package: TUV678, Truck: XX6481-OE
  Package: WXY901, Truck: IM3876-ID
  Package: ZAB234, Truck: YM8145-JL
  Package: CDE567, Truck: GV2229-KA
  Package: FGH890, Truck: JM2670-MP
  Package: IJK123, Truck: UU7511-AZ
  Package: LMN456, Truck: WT6802-VG
  Package: OPQ789, Truck: EN4294-JJ
  Package: RST012, Truck: MZ2817-JD
  Package: UVW345, Truck: EN7090-HC
  Package: XYZ678, Truck: TH5775-AL
  Package: 123ABC, Truck: XT9528-VO
  Package: 456DEF, Truck: BG3586-QX
  Package: 789GHI, Truck: GF1266-WR
  Package: UVW901, Truck: EV8358-AH
  Package: XYZ234, Truck: UM3262-IJ
  Package: 789ABC, Truck: OA5609-FK
  Package: DEF567, Truck: EW9020-EY
  Package: GHI890, Truck: EN7090-HC
  Package: JKL234, Truck: XU3837-CQ
  Package: MNO567, Truck: UW8617-XE
  Package: PQR890, Truck: FZ2594-GD
  Package: STU123, Truck: UW8617-XE
  Package: VWX456, Truck: XX6481-OE
  Package: YZA789, Truck: OV7286-OD
  Package: BCD012, Truck: IM3876-ID
  Package: EFG345, Truck: CD6229-GP
  Package: HIJ678, Truck: UG0812-HJ
  Package: KLM901, Truck: VG5469-RJ
  Package: NOP234, Truck: UK9066-BZ
  Package: QRS567, Truck: YC9978-HS
  Package: TUV890, Truck: OX6875-BZ
```



The script reads input data from trucks.csv file containing information about trucks and packages. Each truck has a name and a maximum capacity, while each package has a name, weight, and distance. The algorithm randomly assigns packages to trucks, evaluates the fitness of each assignment based on the maximum load of trucks, and iteratively evolves the population through selection, crossover, and mutation operations.

The results section presents two solutions: the 'Normal Solution' obtained from the initial population, and the 'Best Solution' obtained after running the genetic algorithm. Each solution lists the assigned packages and their corresponding trucks, providing insights into the optimized assignments.

For instance, the 'Normal Solution' shows a set of package-truck assignments with a total cost of 2467.00. Meanwhile, the 'Best Solution' presents an improved set of assignments with a reduced cost of 1986.00, indicating the algorithm's effectiveness in optimizing the package-truck assignments for more efficient transportation.

Limitations/Difficulties

Decision Tree Model Limitations:

The Decision Tree model, while serving as a simple and interpretable algorithm, has inherent limitations. Overfitting is a concern, as the model might capture noise from the training data, leading to poor generalization on unseen data. Additionally, its limited expressiveness might struggle to capture complex relationships within the dataset. Sensitivity to variations in the training data and a lack of detailed interpretability in explaining specific predictions further constrain its effectiveness.

Random Forest Model Limitations:

The Random Forest model, an ensemble of Decision Trees, offers improvements but introduces its own set of considerations. Its increased complexity may pose challenges in terms of computational intensity and interpretation compared to a single Decision Tree. Proper hyperparameter tuning is essential to optimize performance, and while it mitigates overfitting, the model may still be sensitive to noisy data. The ensemble's black-box nature could limit the depth of interpretability, necessitating a balance between accuracy and model transparency.

K-Means Limitations:

The K-Means clustering code for route optimization has several limitations and constraints that should be acknowledged. Firstly, K-Means is sensitive to the initial placement of centroids, and different initializations may yield varied results. Additionally, the algorithm assumes spherical and equally sized clusters, which may not align with the actual data distribution. The dependence on Euclidean distance for similarity measurement can be problematic if features have different scales or if the data does not adhere to Euclidean geometry. Choosing the optimal number of clusters (K) is subjective, and the elbow method or silhouette analysis used for selection may not always provide definitive answers. Furthermore, K-Means is sensitive to outliers, and the code's one-hot encoding of categorical variables may impact dimensionality and interpretability.

Simple Genetic Algorithm Limitations:

Simple Genetic Algorithm (SGA) for optimizing package-truck assignments has several limitations and constraints. One notable limitation is the reliance on random initialization for the initial population, which may lead to suboptimal solutions, especially for complex problem instances. Additionally, the algorithm's effectiveness heavily depends on the chosen parameters, such as population size, mutation rate, and crossover rate, and finding the optimal values for these parameters can be challenging.

Furthermore, the algorithm assumes a fixed set of trucks and packages, neglecting real-world factors such as dynamic changes in the availability of trucks or the introduction of new packages. This lack of adaptability to changing conditions restricts the algorithm's applicability in dynamic environments. The algorithm also does not consider external factors like traffic conditions, road closures, or time constraints, which are crucial in real-world logistics scenarios.

Conclusions

In conclusion, our approach to problem-solving using machine learning models has yielded successful results, despite acknowledging certain limitations. Our utilization of decision trees and random forest classifiers for route optimization in mining operations has yielded insightful results. Decision trees, while inherently interpretable and versatile, displayed lower accuracy scores, prompting further consideration of additional relevant features for improvement. On the other hand, random forests, with their ensemble nature, exhibited a notable accuracy improvement compared to decision trees. Although challenges persist, such as the need for feature exploration and hyperparameter tuning, the models have laid a solid foundation for ongoing refinement. Continuous collaboration and adaptation will be pivotal as we strive to enhance the efficiency of route planning in mining operations.

The implementation of clustering techniques, particularly K-Means clustering, proved instrumental in suggesting Geographic-based Route Planning to the company. By utilizing the identified clusters, we recommended grouping and assigning routes based on geographic proximity. This strategic approach aims to optimize travel distances and times between stops, potentially improving overall operational efficiency.

The Simple Genetic Algorithm (SGA) showcased its effectiveness in optimizing package-truck assignments. The algorithm's ability to evolve and adapt, as well as its iterative nature, allowed for the identification of improved solutions for the logistics optimization problem. However, it's important to note that the SGA has its own set of limitations, including sensitivity to parameters and the lack of consideration for dynamic factors in real-world scenarios.

In summary, our machine learning solutions have provided valuable insights and recommendations, offering a foundation for further refinement and exploration of advanced techniques to address the challenges presented by route optimization and logistics planning.