Here is a detailed explanation of your code, structured like a report for easy copying:

**Titanic Survival Prediction Code Explanation**

**Code Import and Data Loading**

**Code Snippet:**

import pandas as pd

from sklearn.impute import KNNImputer

**How it Works:**

* pandas is imported to handle data operations.
* KNNImputer from sklearn.impute is imported for handling missing values using K-Nearest Neighbors imputation.

**Why it is Used:**

* pandas is necessary for reading, cleaning, and manipulating datasets.
* KNNImputer is used to fill missing numerical values using a machine-learning approach instead of simple mean or median imputation.

**Code Snippet:**

df = pd.read\_csv(r"C:\Users\Admin\Desktop\all data\4th\Task 2\train.csv")

df.head()

**How it Works:**

* This loads the dataset from the CSV file into a DataFrame named df.
* df.head() displays the first five rows of the dataset.

**Why it is Used:**

* Loading the dataset is the first step in working with data.
* df.head() is useful to inspect the structure of the dataset.

**Handling Duplicates**

**Code Snippet:**

df.duplicated().sum()

**How it Works:**

* This checks for duplicate rows in the dataset and returns the total count of duplicate entries.

**Why it is Used:**

* Duplicate data can skew analysis and model training, so identifying and removing them is important.

**Handling Missing Values**

**Code Snippet:**

impute = KNNImputer()

for i in df.select\_dtypes(include = 'number').columns:

df[i] = impute.fit\_transform(df[[i]])

**How it Works:**

* KNNImputer() is initialized to fill missing values.
* The code loops through all numerical columns and replaces missing values using K-Nearest Neighbors imputation.

**Why it is Used:**

* Instead of dropping missing values or using a simple mean, KNNImputer predicts missing values based on similar data points.

**Encoding Categorical Variables**

**Code Snippet:**

from sklearn.preprocessing import LabelEncoder

le = LabelEncoder()

for i in df.select\_dtypes(include = 'object').columns:

df[i] = le.fit\_transform(df[i])

**How it Works:**

* LabelEncoder is used to convert categorical columns into numerical values.
* The loop encodes all categorical columns.

**Why it is Used:**

* Machine learning models require numerical data, so categorical values must be converted.

**Dropping Unnecessary Columns**

**Code Snippet:**

df = df.select\_dtypes(include = 'number').astype('int64')

columns\_to\_drop = ['Name', 'Cabin', 'Ticket']

df\_cleaned = df.drop(columns=columns\_to\_drop)

**How it Works:**

* Converts numerical columns to integer format.
* Drops unnecessary columns (Name, Cabin, Ticket).

**Why it is Used:**

* Name, Cabin, and Ticket do not contribute to survival prediction.
* Converting data to integers ensures compatibility with machine learning models.

**Splitting Data for Training**

**Code Snippet:**

x = df.drop(columns = ['Survived'])

y = df['Survived']

**How it Works:**

* x contains all features except the target variable (Survived).
* y contains only the target variable (Survived).

**Why it is Used:**

* Separating features and the target variable is required for training a machine learning model.

**Saving Cleaned Data**

**Code Snippet:**

df\_cleaned.to\_csv('train.csv', index=False)

**How it Works:**

* Saves the cleaned dataset to a new CSV file named train.csv.

**Why it is Used:**

* Saving the cleaned data ensures it can be reused without needing to reprocess it each time.

**Loading Train and Test Data**

**Code Snippet:**

train\_data = pd.read\_csv(r'C:\Users\Admin\Desktop\all data\4th\Task 2\train.csv')

test\_data = pd.read\_csv(r'C:\Users\Admin\Desktop\all data\4th\Task 2\test.csv')

**How it Works:**

* Loads the cleaned training dataset (train.csv) and test dataset (test.csv).

**Why it is Used:**

* The model needs separate training and test datasets for evaluation.

**Building the Machine Learning Model**

**Code Snippet:**

from sklearn.ensemble import RandomForestClassifier

**How it Works:**

* Imports the RandomForestClassifier from sklearn.

**Why it is Used:**

* Random Forest is a powerful ensemble learning method that works well for classification problems.

**Selecting Features**

**Code Snippet:**

features = ["Pclass", "SibSp", "Parch"]

X = pd.get\_dummies(train\_data[features])

X\_test = pd.get\_dummies(test\_data[features])

**How it Works:**

* Selects important features (Pclass, SibSp, Parch).
* Converts categorical features into numerical values using pd.get\_dummies().

**Why it is Used:**

* Feature selection improves model efficiency and accuracy.
* pd.get\_dummies() ensures categorical variables are correctly encoded.

**Training the Model**

**Code Snippet:**

model = RandomForestClassifier(n\_estimators=100, max\_depth=5, random\_state=1)

model.fit(X, y)

**How it Works:**

* Initializes a Random Forest model with 100 trees and a maximum depth of 5.
* Trains the model using the training dataset (X, y).

**Why it is Used:**

* Random Forest provides a balance between accuracy and efficiency.
* Limiting depth prevents overfitting.

**Making Predictions**

**Code Snippet:**

predictions = model.predict(X\_test)

print("Run")

**How it Works:**

* Uses the trained model to predict survival for the test dataset (X\_test).
* Prints "Run" to indicate successful execution.

**Why it is Used:**

* Predictions are needed for submission to competitions like Kaggle’s Titanic dataset.

**Saving Predictions**

**Code Snippet:**

output = pd.DataFrame({'PassengerId': test\_data.PassengerId, 'Survived': predictions})

output.to\_csv('submission.csv', index=False)

print("Your submission was successfully saved!")

**How it Works:**

* Creates a DataFrame with PassengerId and Survived predictions.
* Saves the predictions to submission.csv.
* Prints confirmation that the file has been saved.

**Why it is Used:**

* Creating a submission file is necessary for evaluating predictions on unseen data.

This is the complete explanation of your Titanic prediction code. You can copy it directly into a Word document. Let me know if you need any modifications! 🚀