

Machine Learning Assignment Code

Name: Noura Hamoud Alharbi
ID: 222000773

```
import warnings
warnings.filterwarnings("ignore")

import numpy as np
import pandas as pd
from sklearn.compose import ColumnTransformer
from sklearn.feature_selection import RFE, SelectKBest, f_classif
from sklearn.impute import SimpleImputer
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, classification_report,
f1_score
from sklearn.model_selection import GridSearchCV, train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import OneHotEncoder, StandardScaler
from sklearn.tree import DecisionTreeClassifier

# 1. Load dataset (Kaggle dataset should be downloaded and named as below)
file_path = "updated_students.csv"
df = pd.read_csv(file_path)

# 2. Handle missing values and prepare the datasets
df["high_gpa"] = np.where(df["Gpa"] >= 3.0, 1, 0)
X = df.drop(columns=["Gpa", "high_gpa", "ID"])
y = df["high_gpa"]

categorical_features =
X.select_dtypes(include=["object"]).columns.tolist()
numerical_features = X.select_dtypes(exclude=["object"]).columns.tolist()

numeric_transformer = Pipeline(steps=[
    ("imputer", SimpleImputer(strategy="median")),
    ("scaler", StandardScaler())
])

categorical_transformer = Pipeline(steps=[
    ("imputer", SimpleImputer(strategy="most_frequent")),
    ("onehot", OneHotEncoder(handle_unknown="ignore"))
])

preprocessor = ColumnTransformer(transformers=[
    ("num", numeric_transformer, numerical_features),
```

```

        ("cat", categorical_transformer, categorical_features)
    ])

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42, stratify=y)

# 3. Apply machine learning models with Grid Search optimizer
models = {
    "Logistic Regression": {
        "model": LogisticRegression(max_iter=1000),
        "params": {"classifier__C": [0.1, 1]}
    },
    "KNN": {
        "model": KNeighborsClassifier(),
        "params": {
            "classifier__n_neighbors": [5, 7],
            "classifier__weights": ["uniform", "distance"]
        }
    },
    "Decision Tree": {
        "model": DecisionTreeClassifier(random_state=42),
        "params": {"classifier__max_depth": [3, 5, None]}
    }
}

for model_name, config in models.items():
    pipeline = Pipeline(steps=[
        ("preprocessor", preprocessor),
        ("classifier", config["model"])
    ])
    grid_search = GridSearchCV(pipeline, config["params"], cv=3,
scoring="accuracy")
    grid_search.fit(X_train, y_train)
    y_pred = grid_search.predict(X_test)

# 4. Use two feature selection methods (SelectKBest and RFE)
X_train_processed = preprocessor.fit_transform(X_train)
X_test_processed = preprocessor.transform(X_test)

kbest = SelectKBest(score_func=f_classif, k=5)
X_train_kbest = kbest.fit_transform(X_train_processed, y_train)
X_test_kbest = kbest.transform(X_test_processed)

rfe = RFE(estimator=LogisticRegression(max_iter=1000),
n_features_to_select=5)
X_train_rfe = rfe.fit_transform(X_train_processed, y_train)
X_test_rfe = rfe.transform(X_test_processed)

# 5. Apply models to selected features and evaluate
final_model = LogisticRegression(max_iter=1000)

```

```
final_model.fit(X_train_kbest, y_train)  
final_model.fit(X_train_rfe, y_train)
```