Predicting school failure and dropout by using data mining techniques

Snehasish Sen

Data Warehousing and Data Mining

PEC-IT602B

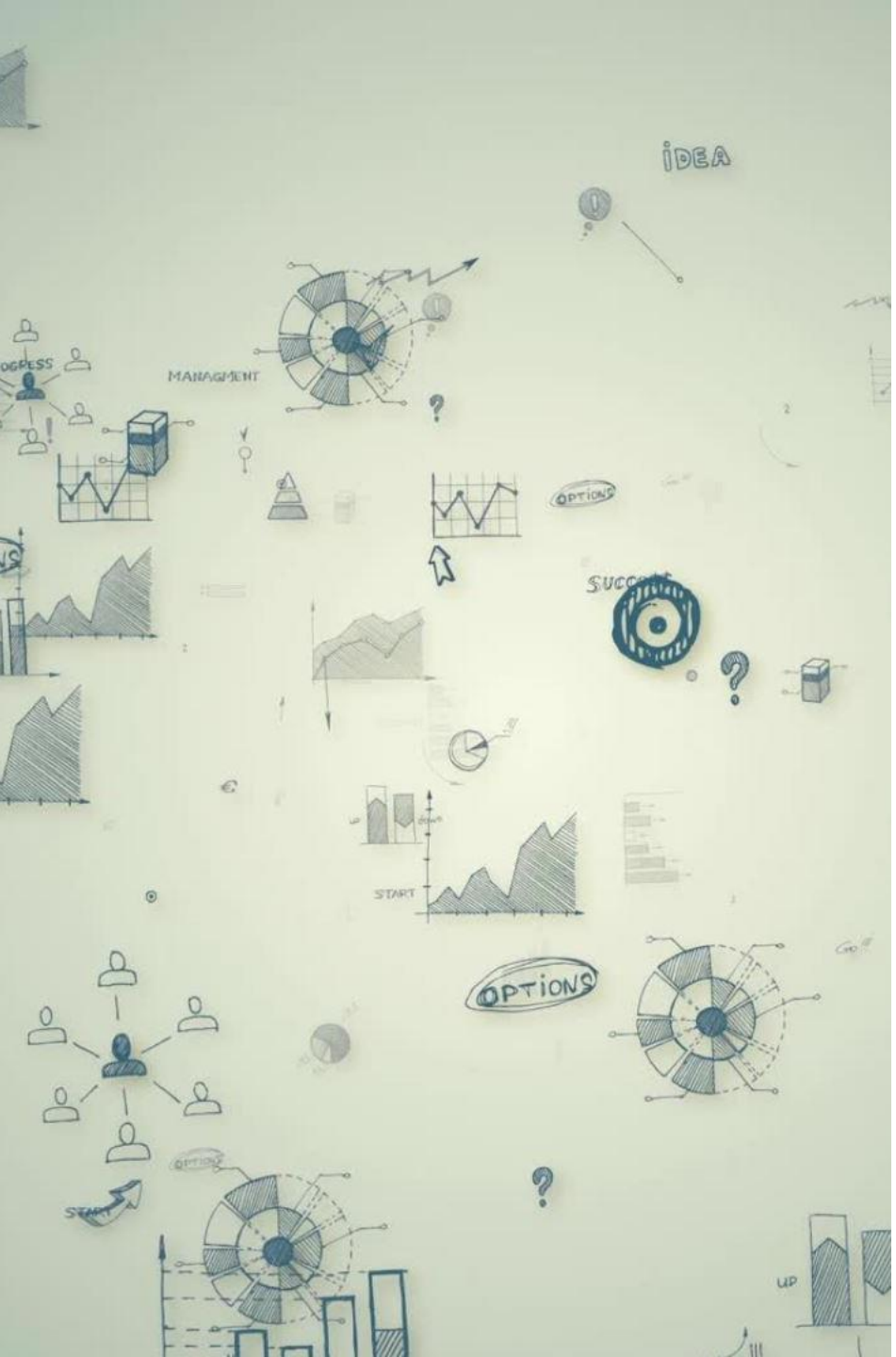
Information Technology

12000221040

Contents:

Introduction

- Data mining is the transformative process of extracting valuable patterns, insights, and knowledge from vast sets of data. Rooted in statistical and artificial intelligence methodologies, data mining enables the discovery of hidden relationships and trends within complex datasets.



Objectives

- Optimize Predictive Accuracy
- Enhance Early Intervention
- Refine Model Robustness
- Explore Feature Importance
- Address Dimensionality

Data Sources

Dataset is obtain from "Kaggle" named **Student Dropout Analysis for School Education by JeevaBharathi.**

Some context about the dataset-->

Comprehensive Student Profile

Predictors of Dropout

Academic Performance Evaluation

Economic Context Analysis

Interdisciplinary Insights

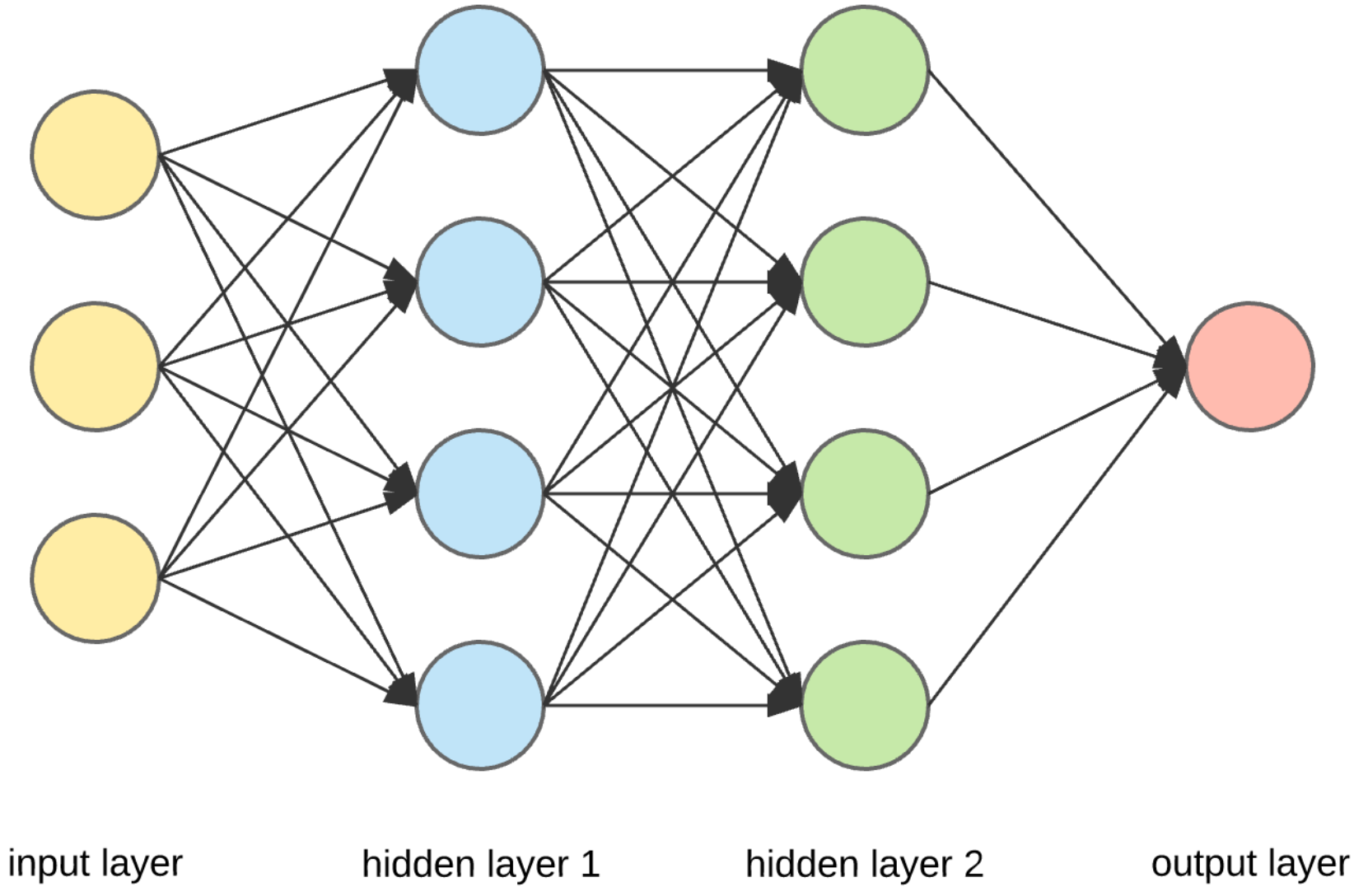
The dataset consists of 35 columns

Column Name	Description
Marital status	The marital status of the student. (Categorical)
Application mode	The method of application used by the student. (Categorical)
Application order	The order in which the student applied. (Numerical)
Course	The course taken by the student. (Categorical)
Daytime/evening attendance	Whether the student attends classes during the day or in the evening. (Categorical)
Previous qualification	The qualification obtained by the student before enrolling in higher education. (Categorical)
Nationality	The nationality of the student. (Categorical)
Mother's qualification	The qualification of the student's mother. (Categorical)
Father's qualification	The qualification of the student's father. (Categorical)
Mother's occupation	The occupation of the student's mother. (Categorical)
Father's occupation	The occupation of the student's father. (Categorical)
Displaced	Whether the student is a displaced person. (Categorical)
Educational special needs	Whether the student has any special educational needs. (Categorical)
Debtor	Whether the student is a debtor. (Categorical)
Tuition fees up to date	Whether the student's tuition fees are up to date. (Categorical)
Gender	The gender of the student. (Categorical)
Scholarship holder	Whether the student is a scholarship holder. (Categorical)
Age at enrollment	The age of the student at the time of enrollment. (Numerical)
International	Whether the student is an international student. (Categorical)
Curricular units 1st sem (credited)	The number of curricular units credited by the student in the first semester. (Numerical)
Curricular units 1st sem (enrolled)	The number of curricular units enrolled by the student in the first semester. (Numerical)
Curricular units 1st sem (evaluations)	The number of curricular units evaluated by the student in the first semester. (Numerical)
Curricular units 1st sem (approved)	The number of curricular units approved by the student in the first semester. (Numerical)

Different Data Mining Techniques

- **Classification**
- **Clustering**
- **Regression**
- **Association Rule Mining**
- **Anomaly Detection**
- **Sequential Pattern Mining**
- **Text Mining (Natural Language Processing)**
- **Neural Networks (Deep Learning)**
- **Dimensionality Reduction**
- **Ensemble Methods**

Model Training



Data Pre-processing

```
dataset['Target'] = dataset['Target'].map({
    'Dropout':0,
    'Enrolled':1,
    'Graduate':2
})

X = dataset.drop("Target", axis=1)
y = dataset['Target']

dataset.head()

from sklearn.model_selection import train_test_split

X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.2,random_state=0)

#using feature scaling

from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test=sc.transform(X_test)

from tensorflow.keras.utils import to_categorical
y_train_encoded = to_categorical(y_train, num_classes=3)
y_test_encoded = to_categorical(y_test, num_classes=3)
```

Using GridSearch to find the best parameters

```
from sklearn.model_selection import GridSearchCV
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble  import RandomForestClassifier

# param_grid = {
#     'n_estimators':[100, 200, 300],
#     'criterion': ['gini', 'entropy', 'log_loss'],
#     'max_features': ['sqrt', 'log2'],
#     'ccp_alpha': [0.1, .01, .001],
#     'max_depth' : [5, 6, 7, 8, 9],
# }

param_grid = {
    'n_estimators':[400],
    'criterion':["gini", "entropy", "log_loss"],
    'min_samples_split':[2, 5, 10, 20],
    'min_samples_leaf':[1],
    'max_features':["sqrt", "log2"],
    'max_depth':[30, 50, 70, 100],
    'bootstrap':[False],
    'random_state':[9],

    'n_jobs':[4, 10],
}

grid = GridSearchCV(RandomForestClassifier(), param_grid=param_grid, refit = True, cv=5, verbose = 3)

# fitting the model for grid search
grid.fit(X_train, y_train)

# print best parameter after tuning
print(grid.best_params_)


# print how our model looks after hyper-parameter tuning
print(grid.best_estimator_)
```

Using RandomForest Classifier

```
rf = RandomForestClassifier(
    bootstrap=False,
    max_depth=30,
    min_samples_split=5,
    n_estimators=400,
    n_jobs=4,
    random_state=9
)
rf.fit(X_train, y_train)
rf.score(X_test, y_test)
```

Validation Accuracy --> 0.8045197740112995 or 80% (best parameters)

Creation and implementation of my ANN model

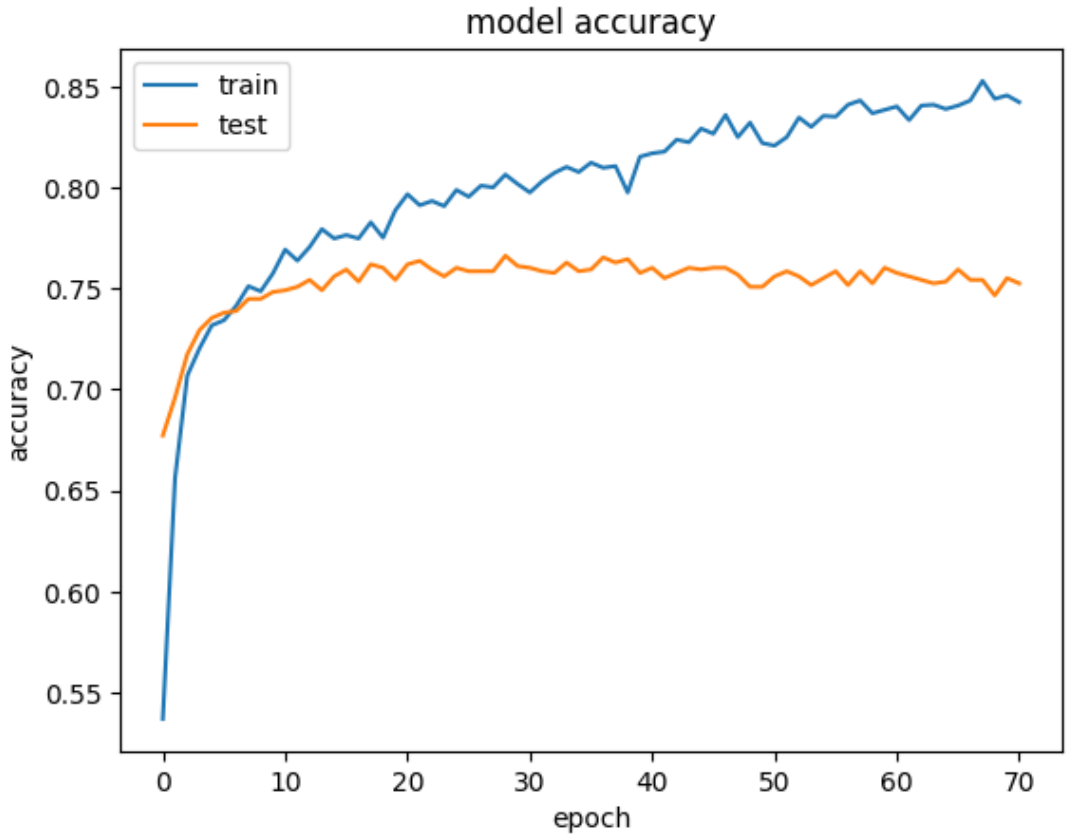
```
##creating the ANN

from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
from tensorflow.keras.layers import LeakyReLU,PReLU,ELU,ReLU
from tensorflow.keras.layers import Dropout

## Input layer, i have 35 input layers
##ANN

classifier = Sequential()
classifier.add(Dense(32,input_shape=(34,),activation='relu'))
classifier.add(Dense(units=26,activation = 'relu'))
classifier.add(Dropout(0.3))
classifier.add(Dense(units=26,activation = 'relu'))
classifier.add(Dropout(0.3))
classifier.add(Dense(3,activation = 'softmax'))

classifier.compile(optimizer='adam', loss='categorical_crossentropy',metrics=['accuracy'])
print(classifier.summary())
```

Creation and implementation of my ANN model

```
##creating the ANN

from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
from tensorflow.keras.layers import LeakyReLU,PReLU,ELU,ReLU
from tensorflow.keras.layers import Dropout

## Input layer, i have 35 input layers
##ANN

classifier = Sequential()
classifier.add(Dense(32,input_shape=(34,),activation='relu'))
classifier.add(Dense(units=26,activation = 'relu'))
classifier.add(Dropout(0.3))
classifier.add(Dense(units=26,activation = 'relu'))
classifier.add(Dropout(0.3))
classifier.add(Dense(3,activation = 'softmax'))

classifier.compile(optimizer='adam', loss='categorical_crossentropy',metrics=['accuracy'])
print(classifier.summary())
```

Model Training Code

```
from sklearn.linear_model import Lasso
lasso_model = Lasso()
lasso_model.fit(X_train, y_train_encoded)
test_predictions = lasso_model.predict(X_test)
train_predictions = lasso_model.predict(X_train)


import tensorflow as tf


early_stopping=tf.keras.callbacks.EarlyStopping(
    monitor="val_loss",
    min_delta=0.0001,
    patience=50,
    verbose=1,
    mode="auto",
    baseline=None,
    restore_best_weights=False,
    start_from_epoch=0,
)
model_history =
classifier.fit(X_train,y_train_encoded,validation_split=0.33,batch_size=32,epochs=200,callb
acks=early_stopping)
```



Benefits and Challenges

- `plt.plot(model_history.history["accuracy"])`
- `plt.plot(model_history.history["val_accuracy"])`
- `plt.title("model accuracy")`
- `plt.ylabel("accuracy")`
- `plt.xlabel("epoch")`
- `plt.legend(["train","test"], loc='upper left')`
- `plt.show()`

The presented graph illustrates an optimal model performance, with a minimal 5% differential between accuracy and validation accuracy. While the proposed Artificial Neural Network (ANN) model demonstrated remarkable accuracy of 99.98%, a challenge emerged with the validation accuracy plateauing at 76.2%. The complexity of the model, characterized by 35 dimensions (34 independent and 1 target), necessitated a strategic approach. Utilizing grid search and hyperparameter tuning, a maximal accuracy of 80% was achieved.

```
from sklearn.decomposition import PCA

pca = PCA(n_components=2)

dataset.head()

X_train

data_set=X
scaled_data = sc.transform(data_set)
pca.fit(scaled_data)

dataset.keys()

x_pca=pca.transform(scaled_data)

scaled_data.shape

x_pca.shape

plt.figure(figsize=(8,6))
plt.scatter(x_pca[:,0],x_pca[:,1],d=X['target'])
plt.xlabel('First component')
plt.xlabel('Second component')
```

Conclusion and Future Steps

- In conclusion, the meticulous exploration of the proposed ANN model revealed impressive accuracy but encountered validation challenges. Hyperparameter tuning provided incremental gains, emphasizing the delicate balance in optimizing a high-dimensional model. Striking an equilibrium remains paramount for robust predictive capabilities in this intricate dataset

The best approach for better accuracy should be pre-processing the data once again with decreasing the dimensions using the PCA , it can drastically help in the improvement of the proposed idea and model.

THANK YOU