**INFO 6105: DATA SCIENCE ENGINEERING METHODS AND TOOLS**

**CAPSTONE PROJECT - CAMPUS PLACEMENT PREDICTIONS**

**CONTRIBUTORS:**

KARTHIK SUBRAMANYA – 002857617

LAKSHMI KUMAR – 002887298

**ABSTRACT:**

In today's competitive job market, predicting campus placements accurately is crucial for both students and educational institutions. Leveraging the power of data science and machine learning, our Streamlit application aims to provide insights into the likelihood of a student being placed based on various criteria.

Our application utilizes a dataset comprising 15 columns, including academic performance metrics, personal information, and other relevant factors. Through exploratory data analysis and predictive modeling techniques, we seek to uncover patterns and trends that influence placement outcomes.

**EXPLORATORY DATA ANALYSIS:**

The basic EDA notebook can be found in the below GitHub Link: <https://github.com/karthiks061992/campus-placements>

**STREMLIT APPLICATION:**

The link to the Streamlit Application can be found in the below link: <https://campus-placements-8mvwccfpf5r2sn8wjxnd2s.streamlit.app/>

**SELF-HEALING MECHANISM (ARCHITECTURE)**

**A diagram of a machine

Description automatically generated**

**WHAT DOES THIS APPLICATION CONSIST OF?**

**IMPORTS:** We have imported necessary libraries such as NumPy, Pandas, Streamlit, Matplotlib, Seaborn, Plotly Express, and scikit-learn modules like Label Encoder, Logistic Regression, and GridSearchCV.

**DATA LOADING AND VISUALIZATION:** In this step, we have loaded a dataset named 'Placement\_Data\_Full\_Class.csv' into a Pandas Data Frame. Then, we have performed some exploratory data analysis (EDA) by creating visualizations like pie charts and count plots to understand the distribution of categorical variables like gender and specialization, and their relationship with the target variable 'status' (placed or not placed).

**MODEL TRAINING AND EVALUATION**: Then we have trained a logistic regression model on the dataset to predict placement status based on various features like gender, academic percentages, board of education, work experience, entrance test score, specialization, and MBA percentage. The model has then been hype tuned using GridSearchCV to find the best parameters. The trained model is saved using pickle.

**STREAMLIT USER INTERFACE**: We have created a Streamlit web application where users can input their academic and personal details like gender, academic percentages, board of education, etc. Then, we used the trained model to predict their placement status and display the result on the UI. Users can also contribute their data to the dataset for model retraining.

**ENHANCE INPUT AND PREDICTION FUNCTIONS**: These functions preprocess the input data and make predictions using the trained model. The enhance input function takes in the user input given through the streamlit application and converts it into a structured vector to be fed into the LR model.

**SAVE DATA FUNCTION**: It saves user-contributed data to the dataset and automatically invokes the rerun() and analytics() function to update the user window on the latest feed.

**MAIN FUNCTION**: It orchestrates the Streamlit application by defining the UI components and their functionalities. The whole model is hosted on the streamlit community cloud and is entirely automated into a ***self healing design*** where the entire user play is in a loop.

**RUN ANALYTICS FUNCTION**: It runs the analytics part of the application, displaying visualizations for gender distribution, placed vs. not placed counts based on gender, specialization ratio, and popular majors.

**RERUN MODEL FUNCTION**: It reloads the dataset, retrains the model, and saves the retrained model. This is also known as a design to support the self-healing mechanism of the software through user inputs and contributions.

**CODE EXPLANATION SNIPPETS:**

**# Function to run analytics**def run\_analytics():

**# Read the dataset into a Data Frame**

df = pd.read\_csv('{}/Placement\_Data\_Full\_Class.csv'.format(common\_path))

**# Displaying the first two rows of the Data Frame**

df.head(2)

**EXPLANATION:** This function is responsible for running analytics on the dataset. It reads the dataset 'Placement\_Data\_Full\_Class.csv' into a Pandas Data Frame and displays the first two rows of the Data Frame.

**# Function to retrain the model and save the data into the csv**

def rerun\_model():

print('This block of cell is used to retrigger the modelling ipynb file')

print('Running....')

**# Reading the dataset to use the latest model in the directory**

df = pd.read\_csv('{}/Placement\_Data\_Full\_Class.csv'.format(common\_path))

df.drop(columns='sl\_no', inplace=True)

**# Encoding categorical variables**

le = LabelEncoder()

df['gender'] = le.fit\_transform(df['gender'])

df['ssc\_b'] = le.fit\_transform(df['ssc\_b'])

df['hsc\_b'] = le.fit\_transform(df['hsc\_b'])

df['hsc\_s'] = le.fit\_transform(df['hsc\_s'])

df['degree\_t'] = le.fit\_transform(df['degree\_t'])

df['workex'] = le.fit\_transform(df['workex'])

df['specialisation'] = le.fit\_transform(df['specialisation'])

**# Splitting the dataset into features (X) and target (y)**

new\_df = df.drop(['salary','status'], axis=1)

X = new\_df.iloc[:,:-1]

y = new\_df.iloc[:,-1]

**# Splitting the dataset into training and testing sets**

x\_train, x\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=101)

**# Initializing and training a Logistic Regression model**

log\_it = LogisticRegression(random\_state=32)

log\_it.fit(x\_train, y\_train)

**# Specifying hyperparameters for GridSearchCV**

parameters = {'penalty': ['l1','l2','elasticnet'], 'C': [1,2,3,5,10,20,30,50], 'max\_iter': [100,200,300]}

**# Tuning hyperparameters using GridSearchCV**

log\_it\_grid = GridSearchCV(log\_it, param\_grid=parameters, scoring='accuracy', cv=10)

log\_it\_grid.fit(x\_train, y\_train)

**# Saving the retrained model**

pickle.dump(log\_it\_grid, open('{}/trained\_model.sav'.format(common\_path),'wb'))

return "Model retrained and saved successfully with rerun analytics"

**EXPLANATION:** This function retrains the machine learning model using the entire dataset using the following steps:

* Reads the dataset and encodes categorical variables.
* Splits the dataset into features (X) and target (y).
* Splits the dataset into training and testing sets.
* Initializes and trains a logistic regression model.
* Specifies hyperparameters for GridSearchCV.
* Tunes hyperparameters using GridSearchCV and saves the retrained model using pickle in the specified directory.

**# Streamlit main function**

def main():

st.title("Campus Placement Prediction Software")

st.image('{}/image-1.jpg'.format(common\_path), use\_column\_width=True)

**# User input section for predicting placement**

st.subheader("Want to know where you stand?? ")

gender = st.selectbox("Please choose your gender", ["M", "F"], index=0)

**# Predict placement upon button click**

output=""

if st.button("Rate your chances!"):

output=enhance\_input(gender, ssc\_p, ssc\_b, hsc\_p, hsc\_b, hsc\_s, degree\_p, degree\_t, workex, etest\_p, specialisation, mba\_p)

st.success(output)

**# User input section for contributing data**

st.subheader("Contribute to our software!!")

st.image('{}/image-2.png'.format(common\_path), use\_column\_width=True)

**# Save contributed data upon button click**

op = ""

if st.button("Save Data !"):

op = save\_data(gen, sscpercentage, sscboard, hscpercentage, hscboard, hscsubject, degreepercentage, degreesubjects, workexp, etestpercentage, spec, mbapercentage, status)

rerun\_model()

st.success("Successfully saved the data and retrained the model")

**# Run analytics upon checkbox click**

if st.checkbox("Run Analytics"):

run\_analytics()

st.image('{}/image-3.jpg'.format(common\_path), use\_column\_width=True)

if \_\_name\_\_ == '\_\_main\_\_':

main()

**EXPLANATIONS:** This part of the code defines the main Streamlit application. It sets up the user interface with input fields for predicting placement and contributing data. It also includes buttons to trigger actions like saving data and running analytics. The “main” function orchestrates the layout and functionality of the Streamlit web application.

**SCREENSHOT OF SPYDER ENVIRONMENT:**

A screen shot of a computer program

Description automatically generated

A screen shot of a computer program

Description automatically generated

A screen shot of a computer program

Description automatically generated

A screenshot of a computer program

Description automatically generated

**HOW TO RUN THE APPLICATION LOCALLY:**

1. **Clone the above-mentioned GitHub repository.**
2. **Open the streamlit-machine.py file in your desired IDE.**
3. **Change the common path to system path and install all the requirements specified in the requirements.txt file using pip/yarn.**
4. **Go to the root directory in the terminal and hit streamlit run “your\_py\_file” which opens a local browser window.**
5. **After the application boots up continue to test it.**

**APPLICATION SCREENSHOTS:**

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated