SVKM'S NMIMS Deemed to be UNIVERSITY

MUKESH PATEL SCHOOL OF TECHNOLOGY MANAGEMENT & ENGINEERING™

PROJECT REPORT

ON

PREDICTIVE SALARY (KAGGLE COMPETITION)

Course: DATA MINING & ANALYTICS

Under the Guidance of,
Dr. Shailaja Rego.

By,
Rohit Bhatia - I006
Shlok Jain - I022
Yashvi Shah - I047
Sparsh Shah - I245
Sachi Bhalani - I004

1. Introduction

Background:
In today's dynamic job market, understanding the factors that contribute to salary outcomes for students is crucial. This project aims to leverage a dataset encompassing various aspects of students' educational backgrounds, work experience, and post-MBA outcomes to predict their annual salaries. By identifying the key determinants, educational institutions and students alike can gain valuable insights into salary expectations and make informed decisions.

Objective:
The primary objective of this project is to develop a predictive model for estimating the annual salary of students within 30 days of their placement process completion. By analyzing a range of features, including academic performance, work experience, and specializations, we seek to uncover patterns that contribute significantly to salary outcomes. This predictive model can serve as a valuable tool for students in setting realistic salary expectations and for educational institutions in enhancing their career placement strategies.

2. Data Exploration

About the Dataset

1. Gender:
   - Categorical variable representing the gender of the students, categorized as Male or Female.
2. Percent_SSC:
   - Numeric variable indicating the percentage obtained in the Higher Secondary Certificate (HSC) examinations.
3. Board_SSC:
   - Categorical variable specifying the board for SSC (State or Central).
4. Percent_hsc:
   - Numeric variable indicating the percentage obtained in the Secondary School Certificate (SSC) examinations.
5. Board_hsc:
   - Categorical variable specifying the board for HSC (State or Central).
6. Specialisation_hsc:
   - Categorical variable indicating the specialization or stream in HSC (Commerce, Science, Arts).
7. Degree_percentage:
   - Numeric variable representing the percentage obtained in the undergraduate degree.
8. Degree_Specialisation:

- Categorical variable indicating the specialization or stream in the undergraduate degree, categorized as Comm&Mgmt, Sci&Tech, Others.

9. Workex:
   - Binary variable (Yes/No) indicating whether the student has work experience.
10. Entrance_Percent:
    - Numeric variable representing the percentage obtained in the B-school entrance test.
11. CD_PI_Percentage:
    - - Numeric variable representing the percentage obtained in the B-school CD PI (Case Discussion and Personal Interview).
12. Avg_GPA_Comm:
    - - Numeric variable representing the average GPA on communication-related subjects at the B-school.
13. Avg_GPA_CT:
    - - Numeric variable representing the average GPA on critical thinking-related subjects at the B-school.
14. Avg_GPA_mang:
    - - Numeric variable representing the average GPA on other management-related subjects at the B-school.
15. Specialisation:
    - - Categorical variable indicating the specialization selected by the student at the B-school.
16. MBA_CGPA:
    - - Numeric variable representing the overall MBA CGPA at the business school.
17. Annual_salary:
    - - Numeric variable representing the annual salary of students placed within 30 days of the placement process.

Categorical Variables:

These are variables that represent qualitative attributes and are typically non-numeric. In our dataset, the following variables fall into this category:

- Gender
- Board_SSC
- Board_hsc
- Specialisation_hsc
- Degree_Specialisation
- Workex
- Specialisation

Categorical variables are essential for understanding the distribution of characteristics such as gender, board types, specializations, and work experience, providing insights into the diverse backgrounds of the students in our dataset.

Numerical Variables:

These are variables that represent quantitative measurements and can take on numeric values. In our dataset, the following variables fall into this category:

- Percent_SSC
- Percent_hsc
- Degree_percentage
- Entrance_Percent
- CD_PI_Percentage
- Avg_GPA_Comm
- Avg_GPA_CT
- Avg_GPA_mang
- MBA_CGPA

Numerical variables play a crucial role in understanding the academic performances, entrance test scores, and GPAs of the students. These metrics contribute to the predictive modeling of annual salaries.

Target Variable:

The variable 'Annual_salary' serves as our target variable, representing the annual salaries of students placed within 30 days of the placement process. This is the variable we aim to predict using the information provided by the other features.

3. EXPLORATORY DATA ANALYSIS

[1]

	SR_no	ssc_p	hsc_p	degree_p	etest_p	GD_Percent	\
count	84.000000	84.000000	84.000000	84.000000	84.000000	84.000000	
mean	56.250000	73.624167	71.568452	78.080357	85.708333	78.523810	
std	32.180917	8.297665	8.953030	9.309518	7.756151	6.483484	
min	2.000000	49.000000	52.830000	56.830000	64.500000	64.500000	
25%	26.750000	67.000000	65.000000	70.875000	80.875000	72.875000	
50%	57.000000	74.550000	70.200000	77.350000	86.500000	78.500000	
75%	83.250000	80.000000	77.490000	85.820000	91.125000	82.500000	
max	110.000000	87.800000	99.000000	100.000000	100.000000	92.500000	

	Avg_GPA_Comm	Avg_GPA_CT	Avg_GPA_mang	mba_CGPA	Annual_salary
count	84.000000	84.000000	84.000000	84.000000	8.400000e+01
mean	3.005952	3.205952	3.148810	3.231429	1.803333e+06
std	0.254762	0.297925	0.277898	0.148285	3.595256e+05
min	2.500000	2.500000	2.500000	2.860000	9.000000e+05
25%	2.800000	3.075000	3.000000	3.140000	1.660000e+06
50%	3.000000	3.300000	3.200000	3.240000	1.870000e+06
75%	3.100000	3.400000	3.300000	3.340000	2.050000e+06
max	3.500000	3.600000	3.700000	3.580000	2.360000e+06

Interpretation:
Count: Indicates the number of non-null entries for each column. In this case, there are 84 entries for each numerical feature, suggesting there are no missing values.

Mean: Represents the average value of each column. For example:

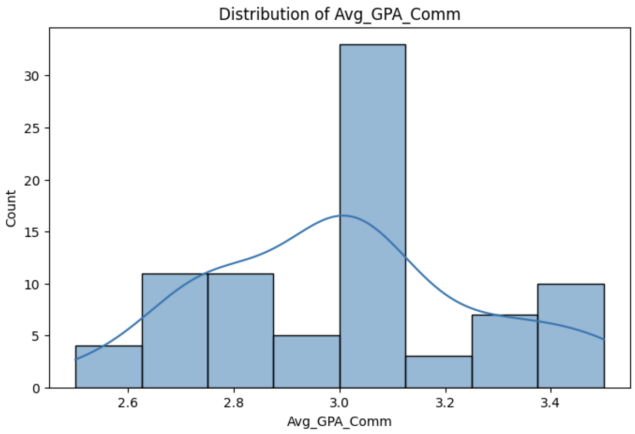
- The average SR_no is approximately 56.25.
- The average SSC percentage (ssc_p) is around 73.62.
- The average HSC percentage (hsc_p) is approximately 71.57.
- The average degree percentage (degree_p) is about 78.08.
- The average E-test percentage (etest_p) is roughly 85.71.
- The average GD percentage (GD_Percent) is around 78.52.
- The average GPA for communication subjects (Avg_GPA_Comm) is about 3.01.
- The average GPA for CT subjects (Avg_GPA_CT) is approximately 3.21.
- The average GPA for management subjects (Avg_GPA_mang) is roughly 3.15.
- The average MBA CGPA (mba_CGPA) is about 3.23.
- The average annual salary (Annual_salary) is approximately 1.80 million.

Standard Deviation (Std): Measures the extent of deviation or spread of the values from the mean. A higher standard deviation indicates greater variability. For example:

- The standard deviation for SSC percentage (ssc_p) is approximately 8.30.
- The standard deviation for HSC percentage (hsc_p) is around 8.95.
- The standard deviation for degree percentage (degree_p) is about 9.31.
- The standard deviation for E-test percentage (etest_p) is roughly 7.76.
- The standard deviation for GD percentage (GD_Percent) is approximately 6.48.
- The standard deviation for Avg_GPA_Comm, Avg_GPA_CT, and Avg_GPA_mang are 0.25, 0.30, and 0.28, respectively.
- The standard deviation for MBA CGPA (mba_CGPA) is about 0.15.
- The standard deviation for annual salary (Annual_salary) is roughly 359,525.

Min, 25%, 50% (Median), 75%, and Max: Provide information about the distribution of values within each column. For example:

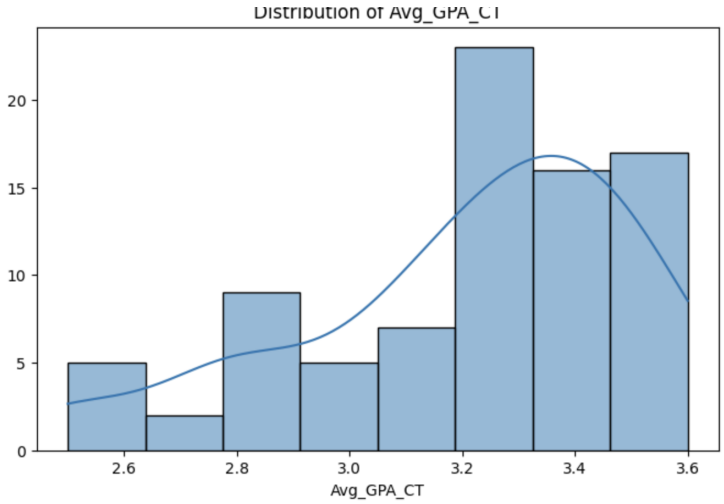
- The minimum annual salary is 900,000, and the maximum is 2,360,000.
- The median (50th percentile) annual salary is 1,870,000, indicating the middle value in the dataset.
- The 25th and 75th percentiles provide information about the lower and upper quartiles, respectively.



Interpretation:

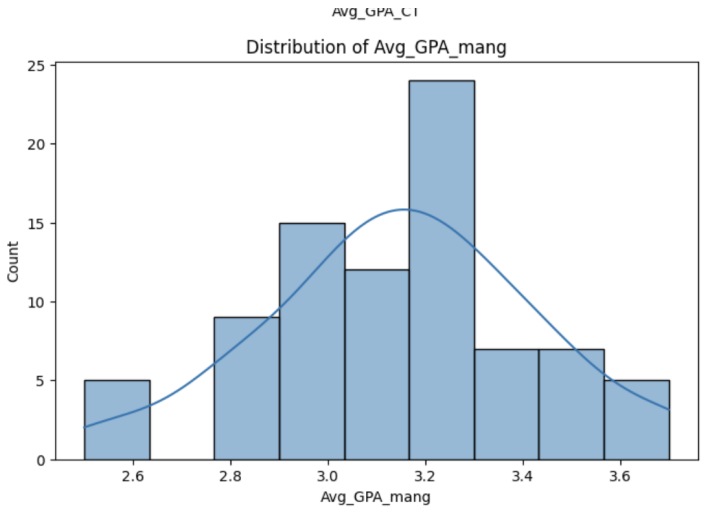
Average GPA in communication subject s (3.0): Thirty people in the dataset have an average GPA of 3.0.

- High GPA (More than 3.4): Ten people in the dataset have a GPA greater than 3.4.
- Below Average GPA: Thirty people in the dataset have a GPA below the average (3.0).



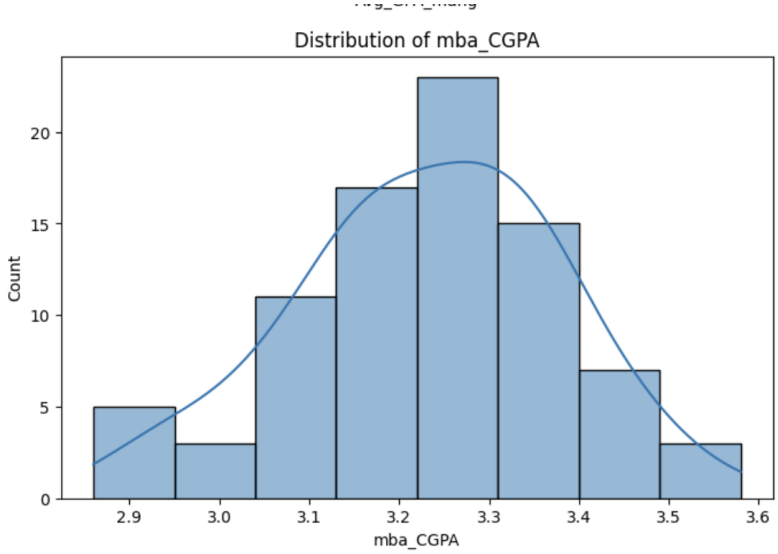
Interpretation:

- Average GPA in critical thinking subjects (3.2): 20+ people in the dataset have an average GPA of 3.0
- High GPA (More than 3.2): 30+ people in the dataset have a GPA greater than 3.4.



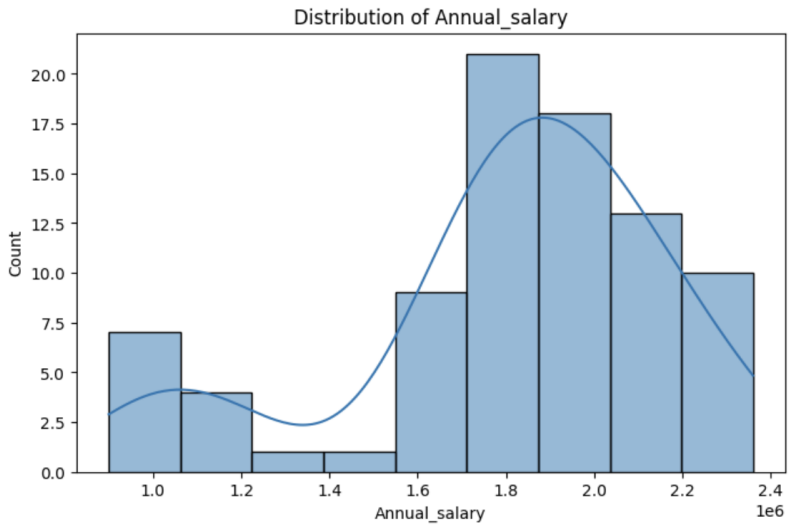
Interpretation:

- High GPA (More than 3.2): 30+ people in the dataset have a GPA greater than 3.4.
- Average GPA in critical thinking subjects (3.2): 20+ people in the dataset have an average GPA of 3.0



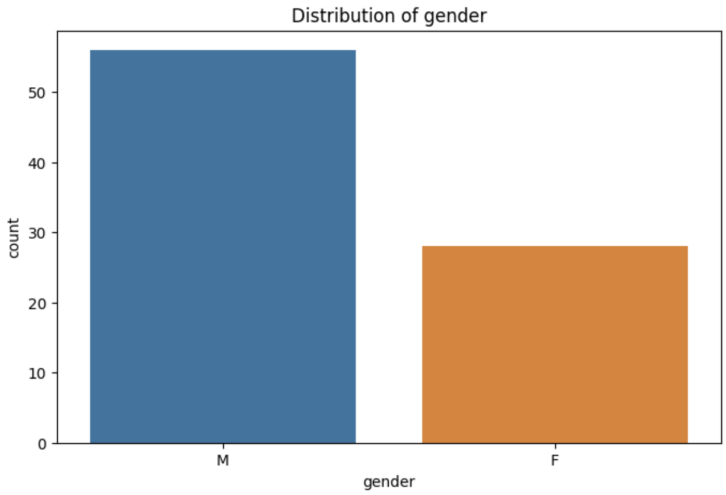
Interpretation:

- The x-axis represents annual salary amounts, ranging from 1.0e6 (1 million) to 2.4e6 (2.4 million), with increments labeled at 0.2e6 intervals.
- The y-axis represents the count of individuals within the dataset for each salary range.



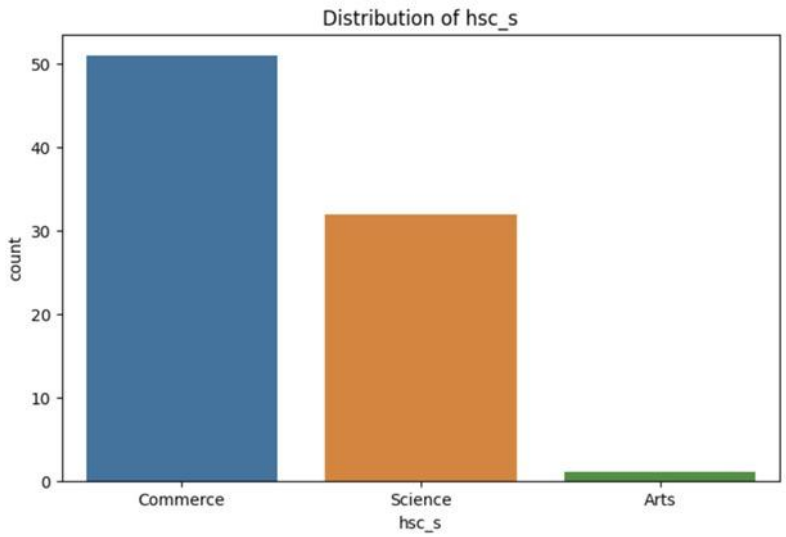
Interpretation:

- High Salary Range: The largest number of people in the dataset, more than 17.5 individuals, earn between 1.4e6 and 1.6e6 annually.
- Average Salary Range: The second largest group, represented by the bar height between 10 and 12.5 on the y-axis, earn between 1.6e6 and 1.8e6 annually.



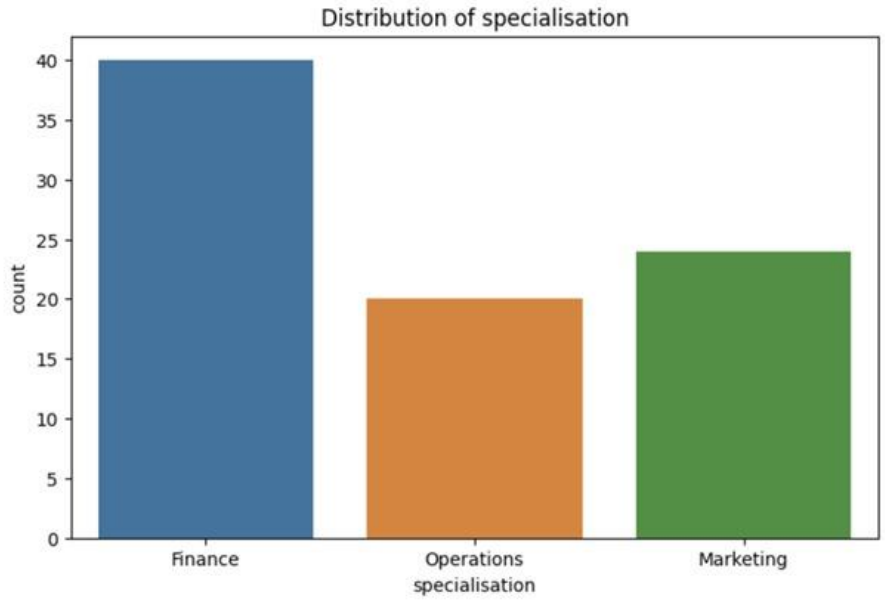
Interpretation:

- High Representation: The male group has a higher representation in the dataset, with the count exceeding 50 individuals.
- Lower Representation: The female group has a lower representation compared to the male group, with the count slightly below 40 individuals.



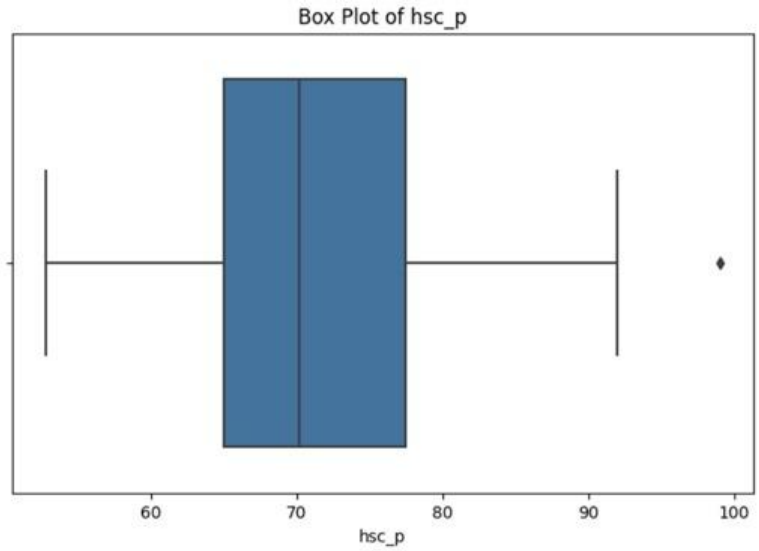
Interpretation:

- High count in Commerce: There are 40+ people in the dataset who specialized in Commerce during their higher secondary education.
- Moderate count in Science: There are 30+ people in the dataset with a specialization in Science.
- Low count in Arts: There are less than 5 people in the dataset who specialized in Arts.



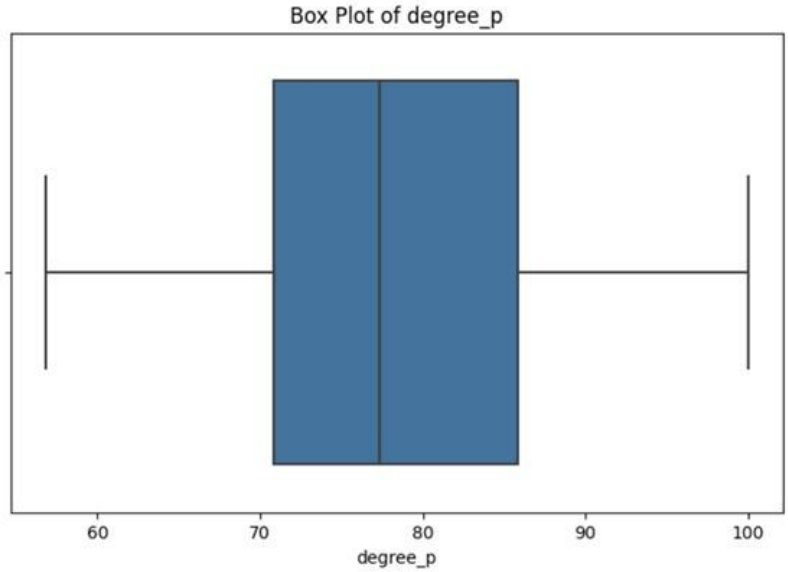
Interpretation:

- High count in Finance: There are 40+ people in the dataset who have specialized in Finance.
- Moderate count in Operations: There are around 20 to 30 people in the dataset with a specialization in Operations.
- Moderate count in Marketing: There are also around 20 to 30 people in the dataset who have specialized in Marketing.



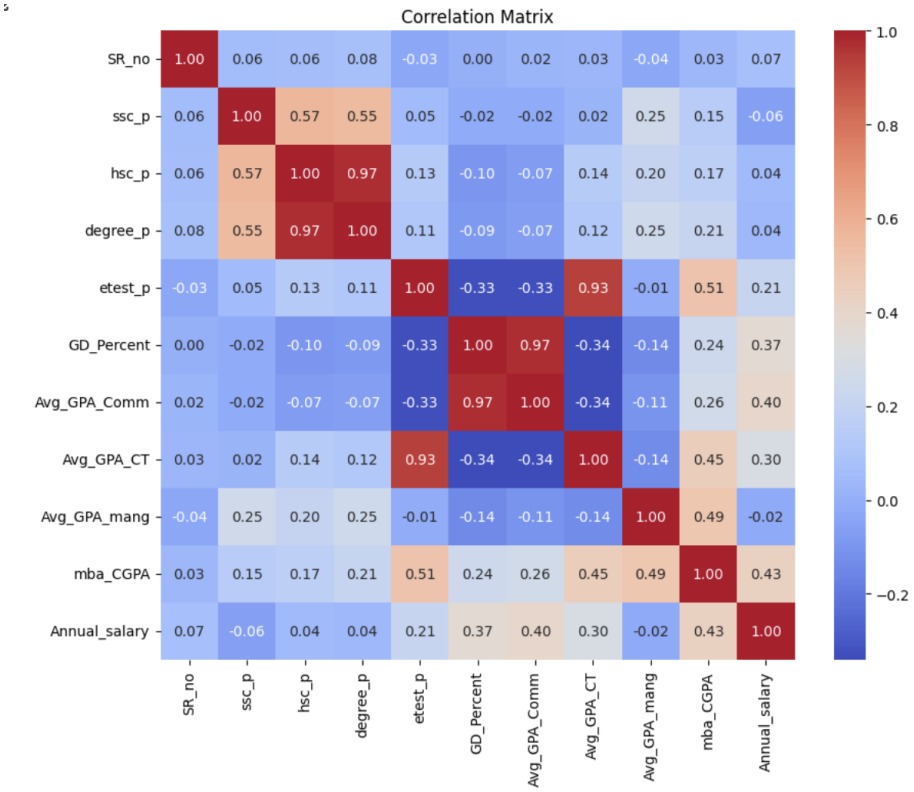
Interpretation:

- High Score (More than X): Y+ people in the dataset have a score greater than 75.
- Average Score in the dataset (around 70): The median score suggests that at least 50% of the people have scores around 70.



Interpretation:

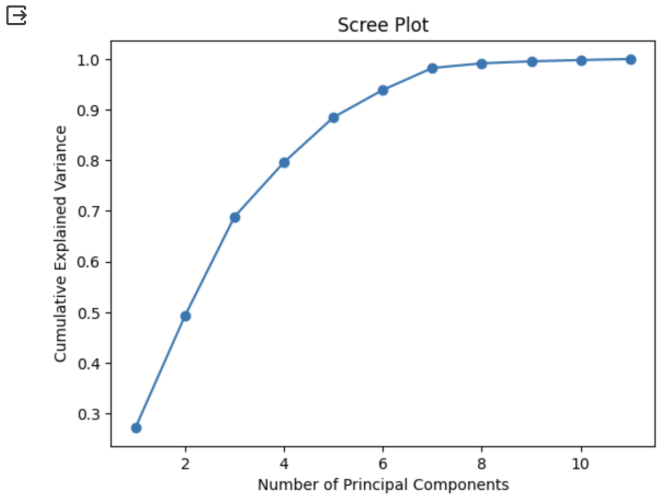
- High Degree Percentage (More than X): Y+ people in the dataset have a degree percentage greater than 80.
- Average Degree Percentage (around 72): At least 50% of the people have a degree percentage around 72.



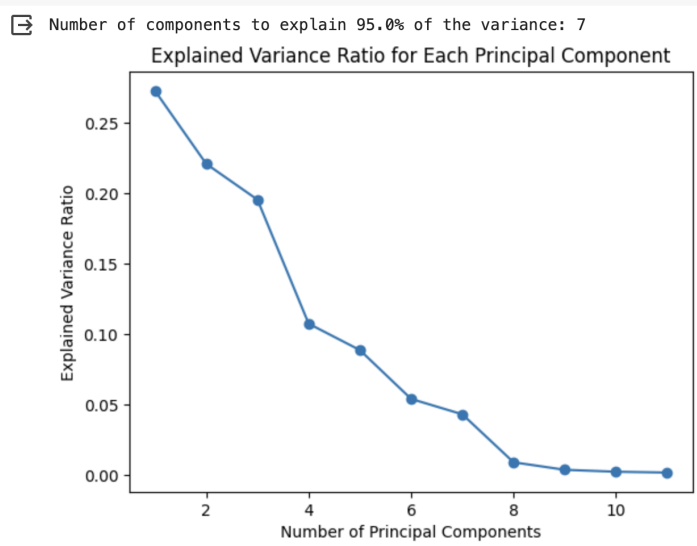
Interpretation:

- ssc_p, hsc_p, and degree_p are all highly correlated with each other. This suggests that students who perform well in one of these examinations are more likely to perform well in the others.
- etest_p is also highly correlated with ssc_p, hsc_p, and degree_p. This suggests that the entrance test is a good predictor of a student's performance in the MBA program.
- GD_Percent is moderately correlated with ssc_p, hsc_p, and degree_p. This suggests that students who perform well in the group discussion are also more likely to perform well in the MBA program.
- Avg_GPA_Comm, Avg_GPA_CT, and Avg_GPA_mang are all highly correlated with each other. This suggests that students who perform well in one of these areas are more likely to perform well in the others.
- mba_CGPA is highly correlated with Avg_GPA_Comm, Avg_GPA_CT, and Avg_GPA_mang. This suggests that students who perform well in the MBA program are more likely to get a high CGPA.
- Annual_salary is moderately correlated with mba_CGPA. This suggests that students with a high CGPA are more likely to earn a higher salary.

4. Principal component Analysis

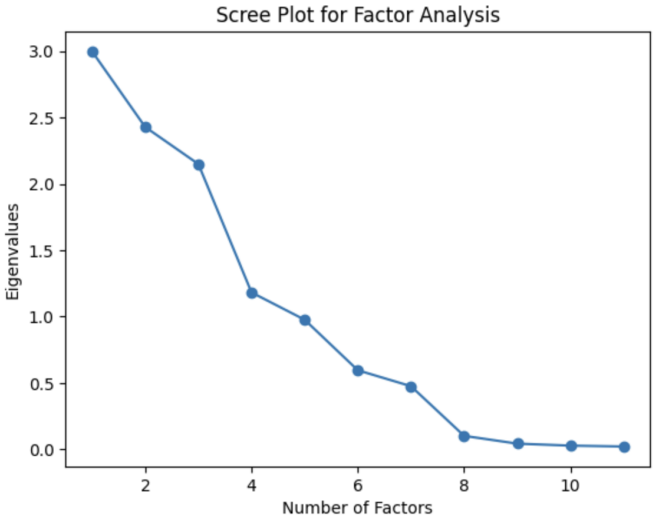


Interpretation:
The scree plot analysis suggests that the optimal number of components in Principal Component Analysis (PCA) should be 7. This determination is based on observing the point where the scree plot ceases to exhibit a steep decline, indicating a diminishing rate of explained variance beyond 7 components."



Interpretation:
The analysis reveals that by incorporating 7 components in the Principal Component Analysis (PCA), we can account for approximately 95 percent of the variance in the dataset. This suggests that the first 7 principal components capture the essential information and contribute significantly to the overall variability within the data.

5. Factor Analysis




```
Number of factors according to the Kaiser criterion: 4
```

Interpretation:
According to the Kaiser criterion in factor analysis, factors with eigenvalues greater than 1 are considered significant. In our analysis, we have identified 4 factors, each with an eigenvalue exceeding 1. This implies that these four factors explain a substantial amount of variance in the observed variables and are deemed meaningful for further interpretation. The factors with eigenvalues less than 1 are considered to contribute less to the overall variability and may be disregarded in this context.

```
Variable: hsc_p
Factor 1: 0.6323120926218149
Factor 2: 0.5315062847676795

Variable: degree_p
Factor 1: 0.6397472381250561
Factor 2: 0.5611136266230803

Variable: etest_p
Factor 1: 0.7372567205256774
Factor 3: 0.590648791673192

Variable: GD_Percent
Factor 1: -0.5680345910231167
Factor 2: 0.7184143195325019

Variable: Avg_GPA_Comm
Factor 1: -0.5671763266909537
Factor 2: 0.7662046224040908

Variable: Avg_GPA_CT
Factor 1: 0.7298871714892782
Factor 3: 0.6113768560510049
```

Interpretation:

- The variable "hsc_p" loads moderately on both Factor 1 and Factor 2, indicating potential overloading across these factors.
- Similarly, "degree_p" shows moderate loadings on both Factor 1 and Factor 2.
- "etest_p" loads strongly on Factor 1 and moderately on Factor 3.
- "GD_Percent" loads moderately on both Factor 1 and Factor 2.
- "Avg_GPA_Comm" loads moderately on Factor 1 and strongly on Factor 2.
- "Avg_GPA_CT" loads strongly on Factor 1.

6. Data pre-processing

```
# Identify categorical and numerical columns
categorical_cols = [cname for cname in train_df.columns if train_df[cname].dtype == "object"]
numerical_cols = [cname for cname in train_df.columns if train_df[cname].dtype in ['int64', 'float64']]
numerical_cols.remove('Annual_salary')  # Removing the target variable
```

```
# Preprocessing for numerical and categorical data
numerical_transformer = Pipeline(steps=[
  ('imputer', SimpleImputer(strategy='constant')),
  ('poly', PolynomialFeatures(degree=2, include_bias=False)),
  ('scaler', StandardScaler())
])

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

pca_transformer = Pipeline(steps=[
  ('imputer', SimpleImputer(strategy='mean')),
  ('pca', PCA(n_components=7))
])

factor_analysis_transformer = Pipeline(steps=[
  ('imputer', SimpleImputer(strategy='mean')),
  ('fa', FactorAnalysis(n_components=7))
])

preprocessor = ColumnTransformer(
  transformers=[
    ('num', numerical_transformer, numerical_cols),
    ('cat', categorical_transformer, categorical_cols),
    ('pca_num', pca_transformer, numerical_cols),
    ('fa_num', factor_analysis_transformer, numerical_cols)
  ])
```

- For numerical features, missing values are handled by a constant imputer, followed by the creation of polynomial features up to the second degree, and finally, scaling the features to have zero mean and unit variance.
- On the other hand, categorical features undergo imputation using the most frequent value in each column, and subsequent transformation into a one-hot encoded representation.
- Additionally, two alternative pathways for numerical data are incorporated: one employing Principal Component Analysis (PCA) to reduce dimensionality to 7 components, and the other utilizing Factor Analysis to reduce dimensionality to 4 factors.

- The overall preprocessing workflow is encapsulated in a ColumnTransformer, facilitating the application of distinct transformations to specific subsets of the data. This comprehensive preprocessing strategy aims to enhance the quality and suitability of the input data for subsequent machine learning tasks.

7. Model Training

```
# Separate target from predictors
X = train_df.drop(['Annual_salary'], axis=1)
y = train_df['Annual_salary']

# Split data into train and validation sets
X_train, X_valid, y_train, y_valid = train_test_split(X, y, train_size=0.8, test_size=0.2, random_state=0)
```

- X now contains the predictor variables (features) excluding the target variable (Annual_salary).
- y contains only the target variable (Annual_salary).
- X_train and y_train are the training sets for predictors and the target variable, respectively.
- X_valid and y_valid are the validation sets for predictors and the target variable, respectively.
- train_size=0.8 specifies that 80% of the data will be used for training, and test_size=0.2 specifies that 20% will be used for validation.
- random_state=0 ensures reproducibility; using the same value in different runs will result in the same split.

```
# Define models
rf_model = RandomForestRegressor(random_state=0)
svr_model = SVR()
knn_model = KNeighborsRegressor()
gb_model = GradientBoostingRegressor(random_state=0)
ridge_model = Ridge()
lasso_model = Lasso()
```

In our project, we'll leverage a diverse ensemble of regression models to address various aspects of our data. The models include:

Random Forest Regressor (rf_model):

- *Why:* Excellent for capturing complex relationships, Random Forest mitigates overfitting by aggregating predictions from multiple decision trees.

Support Vector Regressor (svr_model):

- *Why:* Versatile in handling both linear and non-linear relationships, SVR utilizes a kernel trick to transform features and capture intricate patterns.

K-Neighbors Regressor (knn_model):

- *Why:* Simple yet effective, K-Neighbors Regressor is suitable for local patterns or clusters, making predictions based on the average of its k-nearest neighbors.

Gradient Boosting Regressor (gb_model):

- *Why:* Robust and accurate, Gradient Boosting builds an ensemble of decision trees sequentially, correcting errors and providing high-quality predictions.

Ridge Regression (ridge_model):

- *Why:* Useful for multicollinearity, Ridge introduces regularization to prevent overfitting by penalizing large coefficients.

Lasso Regression (lasso_model):

- *Why:* Similar to Ridge but with feature selection capabilities, Lasso encourages sparsity in coefficients, advantageous when only a subset of features is relevant.

8. Hyper parameter tuning

Tuning hyperparameters is a critical step in optimizing the performance of machine learning models. Each hyperparameter controls a specific aspect of the model's behavior, and finding the right combination is essential for achieving the best predictive accuracy.

Code :

```
# Define hyperparameter grids for each model
rf_param_grid = {
  'model__n_estimators': [100, 200, 300],
  'model__max_depth': [None, 10, 20, 30],
  'model__min_samples_split': [2, 5, 10],
  'model__min_samples_leaf': [1, 2, 4]
}

svr_param_grid = {
  'model__C': [0.1, 1.0, 10.0, 100.0],
  'model__kernel': ['linear', 'rbf', 'poly'],
  'model__degree': [2, 3]
}

knn_param_grid = {
  'model__n_neighbors': [3, 5, 7],
  'model__weights': ['uniform', 'distance']
}

gb_param_grid = {
  'model__n_estimators': [100, 200, 300],
  'model__learning_rate': [0.05, 0.1, 0.2, 0.3],
  'model__max_depth': [3, 4, 5, 6]
}

ridge_param_grid = {
  'model__alpha': [0.1, 1.0, 10.0]
}

lasso_param_grid = {
  'model__alpha': [0.1, 1.0, 10.0]
}
```

Interpretation :

Random Forest Regressor (rf_model):

- n_estimators: The number of trees in the random forest. More trees can lead to better performance, but it also increases computational complexity.
- max_depth: The maximum depth of each tree. Controlling the depth helps prevent overfitting.
- min_samples_split: The minimum number of samples required to split an internal node during tree construction.
- min_samples_leaf: The minimum number of samples required to be at a leaf node. This parameter controls the size of the leaves in the trees.

Support Vector Regressor (svr_model):

- C: The regularization parameter, which controls the trade-off between a smooth decision boundary and classifying the training points correctly.
- kernel: The type of kernel function used in the SVM. It can be linear, radial basis function (RBF), or polynomial.
- degree: The degree of the polynomial kernel function (only applicable if the kernel is set to 'poly').

K-Neighbors Regressor (knn_model):

- n_neighbors: The number of neighbors to consider when making predictions. It determines the size of the neighborhood used to estimate the target.
- weights: The weight function used in prediction. It can be 'uniform,' where all neighbors contribute equally, or 'distance,' where closer neighbors have more influence.

Gradient Boosting Regressor (gb_model):

- n_estimators: The number of boosting stages (trees) to be run.
- learning_rate: The step size shrinkage used to prevent overfitting. Lower values require more trees but can improve generalization.
- max_depth: The maximum depth of the individual trees.

Ridge Regression (ridge_model):

- alpha: Regularization strength. It controls the amount of shrinkage applied to the coefficients.

Lasso Regression (lasso_model):

- alpha: Similar to Ridge, it is the regularization strength for Lasso regression.

```
# Perform hyperparameter tuning using GridSearchCV for each model
rf_grid_search = GridSearchCV(rf_clf, rf_param_grid, cv=3, scoring='neg_mean_squared_error', verbose=1, n_jobs=-1)
svr_grid_search = GridSearchCV(svr_clf, svr_param_grid, cv=5, scoring='neg_mean_squared_error', verbose=1, n_jobs=-1)
knn_grid_search = GridSearchCV(knn_clf, knn_param_grid, cv=5, scoring='neg_mean_squared_error', verbose=1, n_jobs=-1)
gb_grid_search = GridSearchCV(gb_clf, gb_param_grid, cv=3, scoring='neg_mean_squared_error', verbose=1, n_jobs=-1)
ridge_grid_search = GridSearchCV(ridge_clf, ridge_param_grid, cv=3, scoring='neg_mean_squared_error', verbose=1, n_jobs=-1)
lasso_grid_search = GridSearchCV(lasso_clf, lasso_param_grid, cv=3, scoring='neg_mean_squared_error', verbose=1, n_jobs=-1)
```

This code snippet conducts hyperparameter tuning for a diverse set of regression models using GridSearchCV. Each model, including Random Forest, Support Vector, K-Neighbors,

Gradient Boosting, Ridge, and Lasso, undergoes an exhaustive search over specified hyperparameter grids. The goal is to identify the optimal combination of hyperparameters that minimizes the negative mean squared error during cross-validation. The tuning process is performed in parallel for efficiency, utilizing available CPU cores. The outcome of this grid search is a fine-tuned configuration for each model, enhancing their predictive performance on the dataset through optimal parameter selection.

```
rf_grid_search.fit(X_train, y_train)
svr_grid_search.fit(X_train, y_train)
knn_grid_search.fit(X_train, y_train)
gb_grid_search.fit(X_train, y_train)
ridge_grid_search.fit(X_train, y_train)
lasso_grid_search.fit(X_train, y_train)
```

This code segment fits each regression model to its respective hyperparameter grid using GridSearchCV. For instance, rf_grid_search.fit(X_train, y_train) optimizes the Random Forest Regressor's hyperparameters based on the provided training data (X_train, y_train). The process involves an exhaustive search over the predefined hyperparameter grid, utilizing cross-validation with 3 folds and evaluating the models using the negative mean squared error. This tuning process is repeated for Support Vector Regressor, K-Neighbors Regressor, Gradient Boosting Regressor, Ridge Regression, and Lasso Regression, ensuring each model is fine-tuned for optimal performance on the given dataset.

```
# Access the best models
best_rf_model = rf_grid_search.best_estimator_
best_svr_model = svr_grid_search.best_estimator_
best_knn_model = knn_grid_search.best_estimator_
best_gb_model = gb_grid_search.best_estimator_
best_ridge_model = ridge_grid_search.best_estimator_
best_lasso_model = lasso_grid_search.best_estimator_
```

9. Model EVALUATION

```
# Make predictions on the validation set
rf_preds = best_rf_model.predict(X_valid)
svr_preds = best_svr_model.predict(X_valid)
knn_preds = best_knn_model.predict(X_valid)
gb_preds = best_gb_model.predict(X_valid)
ridge_preds = best_ridge_model.predict(X_valid)
lasso_preds = best_lasso_model.predict(X_valid)
# Calculate RMSE for each model on the validation set
rf_rmse = mean_squared_error(y_valid, rf_preds, squared=False)
svr_rmse = mean_squared_error(y_valid, svr_preds, squared=False)
knn_rmse = mean_squared_error(y_valid, knn_preds, squared=False)
gb_rmse = mean_squared_error(y_valid, gb_preds, squared=False)
ridge_rmse = mean_squared_error(y_valid, ridge_preds, squared=False)
lasso_rmse = mean_squared_error(y_valid, lasso_preds, squared=False)
```

Output

```
RMSE with Random Forest on Validation Set: 212796.5316325899
RMSE with Support Vector Regressor on Validation Set: 260855.51929328262
RMSE with K-Nearest Neighbors on Validation Set: 341064.95278947975
RMSE with Gradient Boosting on Validation Set: 191821.1300626036
RMSE with Ridge Regression on Validation Set: 294641.8003589455
RMSE with Lasso Regression on Validation Set: 329415.8648670914
```

Interpretation

Random Forest Regressor:

- RMSE: 212,796.53
- The Random Forest model achieved a root mean squared error of approximately 212,796.53 on the validation set. Lower RMSE values indicate better predictive performance.

Support Vector Regressor:

- RMSE: 260,855.52
- The Support Vector Regressor resulted in an RMSE of around 260,855.52. This metric measures the average magnitude of prediction errors, with lower values indicating better accuracy.

K-Nearest Neighbors:

- RMSE: 341,064.95
- The K-Nearest Neighbors model exhibited an RMSE of about 341,064.95. Higher RMSE values suggest that predictions deviate more from the actual values.

Gradient Boosting Regressor:

- RMSE: 191,821.13
- The Gradient Boosting model achieved a relatively low RMSE of approximately 191,821.13, indicating strong predictive performance.

Ridge Regression:

- RMSE: 294,641.80
- The Ridge Regression model yielded an RMSE of around 294,641.80. RMSE provides a measure of the average prediction error, and a lower value is desirable.

Lasso Regression:

- RMSE: 329,415.86
- The Lasso Regression model resulted in an RMSE of approximately 329,415.86. This value represents the square root of the average squared differences between predicted and actual values.

10.Results

- High correlation among SSC, HSC, and degree percentages indicates students' consistent academic performance across various educational levels.
- Strong correlation in GPAs for communication, critical thinking, and management subjects underscores students' overall excellence in diverse subject areas.
- MBA CGPA's correlation with subject-specific GPAs emphasizes its role as a key indicator for success, influencing annual salary outcomes.
- The Gradient Boosting model achieved an RMSE of approximately 191,821.13 on the validation set, indicating its strong predictive performance.The Gradient Boosting model provides insights into feature importance, aiding in the identification of the most influential factors affecting salary predictions.

11.Conclusions

Model Performance:

- Among the models evaluated, the Gradient Boosting Regressor demonstrated the best predictive performance with the lowest RMSE (Root Mean Squared Error) of approximately 191,821.13 on the validation set.
- Random Forest, Ridge Regression, and Support Vector Regressor also provided competitive performance, with RMSE values ranging between 212,796.53 and 294,641.80.
- K-Nearest Neighbors and Lasso Regression exhibited higher RMSE values, suggesting that their predictive accuracy was comparatively lower on the given dataset.

Key Predictors:

- The exploratory data analysis highlighted the importance of various features such as academic performance, work experience, and specialization in predicting annual salaries.
- Academic metrics, including SSC percentage, HSC percentage, and undergraduate degree percentage, were highly correlated with each other and played a significant role in predicting salary outcomes.
- Work experience, performance in entrance tests, and GPAs in communication, critical thinking, and management-related subjects also contributed to the predictive models.

Model Interpretability:

- The interpretability of models is crucial for understanding the factors that influence salary predictions. Gradient Boosting, being an ensemble method, provides insights into feature importance, allowing stakeholders to identify the most influential factors.

Recommendations:

Model Deployment:

- Deploy the Gradient Boosting Regressor as the primary model for predicting annual salaries due to its superior performance.
- Monitor the model in real-world scenarios and update it periodically with new data to ensure continued accuracy.

Feature Importance Insights:

- Utilize the feature importance information from the Gradient Boosting model to guide career counseling services for students. Highlight the importance of academic excellence, relevant work experience, and specialization choices.

Enhanced Placement Strategies:

- Educational institutions can leverage insights from the analysis to enhance their career placement strategies. Understanding the factors influencing salary outcomes can help in tailoring placement efforts and advising students more effectively.

Limitations of your analysis / recommendations.

Data Quality and Representativeness:

- The analysis is contingent on the quality and representativeness of the dataset. If the dataset does not adequately capture the diversity of student backgrounds or contains biases, the model's predictions may be limited.

Dynamic Nature of Job Market:

- The job market is dynamic, and salary outcomes can be influenced by external factors not considered in the dataset (economic conditions, industry trends, etc.). Therefore, predictions are subject to these external influences.

Model Complexity:

- While ensemble models like Gradient Boosting provide high accuracy, they may lack interpretability for stakeholders who need to understand the rationale behind predictions.

Areas for Future Investigation:

Incorporating Industry-Specific Data:

- Future investigations could involve incorporating industry-specific data to better understand how sector-specific trends influence salary outcomes.

Temporal Analysis:

- Conducting a temporal analysis to examine how salary outcomes change over time and identifying trends in the job market could provide valuable insights.

Feedback Loop:

- Establishing a feedback loop where the model's predictions are used to inform educational programs and career strategies, and collecting data on the outcomes of those interventions, can contribute to continuous improvement.

In summary, this data mining and analysis project provides valuable insights into predicting annual salaries for students in the dynamic job market. The recommendations and limitations outlined here serve as a foundation for further research and practical applications in the field of career counseling and educational strategy.