A project report on

IBM EMPLOYEE ATTRITION & PERFORMANCE -AI ML

Submitted in partial fulfillment for the award of the degree of

B.TECH

by

PAKALA DISHITHA (21BCE7471)

VIT-AP UNIVERSITY

School of Computer Science and Engineering

June, 2024

DECLARATION

I here by declare that the report entitled “IBM EMPLOYEE ATTRITION & PERFORMANCE -AI ML” submitted by me, for the award of the degree of **B.Tech** VIT-AP University is a record of bonafide work carried out by me under the supervision of Dr.Baishalini Sahu.

I further declare that the work reported in this report has not been submitted and will not be submitted, either in part or in full, for the award of any other degree or diploma in this institute or any other institute or university.

P.Dishitha

Place: Amaravati
Date: 26 -04-2024

Signature of team member

Chapter 1

Introduction

1.1 BACKGROUND

IBM, a global technology and consulting company, has been experiencing increased employee turnover and fluctuating performance levels across different departments. Addressing these challenges is crucial for maintaining a productive and stable workforce.

1.2 OBJECTIVE

The aim of addressing employee attrition and performance is to align these efforts with broader organizational goals and values. The primary objective is to reduce employee turnover rates by implementing targeted retention strategies and addressing the factors contributing to voluntary resignations. Additionally, the goal is to retain key talent critical to the organization's success by identifying their specific needs and concerns. Enhancing overall employee satisfaction and engagement levels is also a key objective, achieved through initiatives that focus on work-life balance, career development, and fostering a positive workplace culture. By meeting these objectives, the organization intends to create a more stable and motivated workforce, driving long-term success and sustainability.

1.3 SCOPE

The scope includes data collection and analysis, predictive modeling, and the development of strategies to address attrition and performance issues. This involves collaboration across various departments and levels of the organization. This project encompasses the development of an intelligent AI tool aimed at aiding HR and operations in making informed decisions regarding employee performance and addressing attrition rates. With a focus on reducing turnover rates, retaining key talent, and enhancing overall employee satisfaction, the tool leverages machine learning algorithms to analyze employee data and identify actionable insights. By incorporating features such as performance metrics, sentiment analysis, and predictive modeling, it aims to provide comprehensive support in identifying factors contributing to attrition and strategies for employee retention. Additionally, the tool offers functionalities for monitoring and analyzing trends over time, enabling stakeholders to adapt strategies dynamically. With the potential to optimize workforce management and foster a positive organizational culture, the tool serves as a valuable asset in addressing critical HR challenges and aligning with broader organizational objectives.

Chapter 2

Data Analysis and Visualization

2.1 DATA COLLECTION

The data utilized for this project was collected from Kaggle, a prominent platform for datasets across various domains.

2.2 DATA PREPARATION

Data cleaning is the process of identifying and correcting errors, inconsistencies, and incompleteness in a dataset. It is crucial for ensuring the accuracy and reliability of datafor analysis and decision-making.

Data Understanding: (35*1500)

	Age	Attrition	BusinessTravel	DailyRate	Department	DistanceFromHome	Education	EducationField	EmployeeCount	EmployeeNumber	EnvironmentSatisfaction
0	41	Yes	Travel_Rarely	1102	Sales	1	2	Life Sciences	1	1	2
1	49	No	Travel_Frequently	279	Research & Development	8	1	Life Sciences	1	2	3
2	37	Yes	Travel_Rarely	1373	Research & Development	2	2	Other	1	4	4
3	33	No	Travel_Frequently	1392	Research & Development	3	4	Life Sciences	1	5	4
4	27	No	Travel_Rarely	591	Research & Development	2	1	Medical	1	7	1

Data cleaning processes were implemented to ensure accuracy and reliability, including removing duplicates and handling missing values.

```
IBM_HR.duplicated()
```

```
0       False
1       False
2       False
3       False
4       False
        ...
1995     True
1996     True
1997     True
1998     True
1999     True
Length: 2000, dtype: bool
```

There are duplicate values in the dataset.

```
IBM_HR.drop_duplicates(inplace=True)
```

We dropped the duplicate values.

2.3 DATA VISUALIZATION

Data visualization in Power BI plays a crucial role in transforming raw data into meaningful insights and actionable information. Power BI simplifies data analysis with intuitive visualizations and advanced analytics, enabling informed decision-making and driving business success.

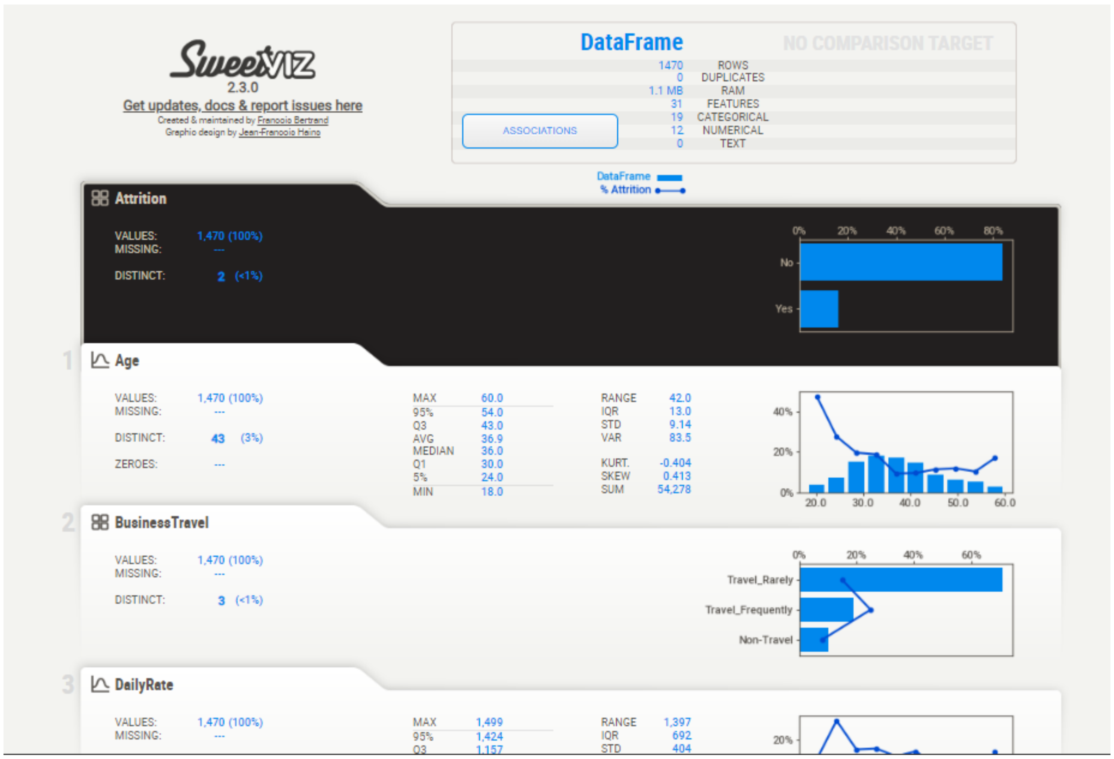
2.4 STATISTICAL ANALYSIS

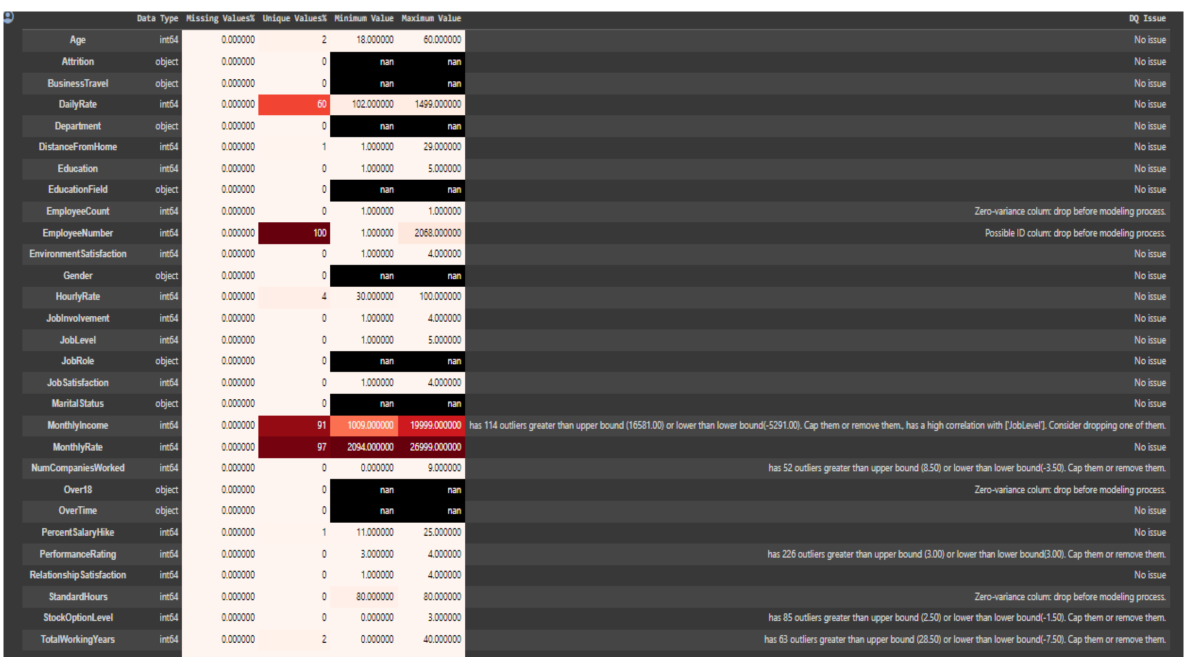
Z-test, a statistical method, can be employed uniquely to identify outliers within a dataset, aiding in pinpointing data points deviating significantly from the mean and assessing their impact on overall analysis and interpretations.

```
[ ] #Z-Test
    scaler = StandardScaler()
    numerical_features = scaler.fit_transform(numerical_features)
    z_scores = stats.zscore(numerical_features)
    abs_z_scores = np.abs(z_scores)
    threshold = 3
    outliers = (abs_z_scores > threshold).all(axis=1)
    print(outliers)

    [False False False ... False False False]
```

2.5 AUTO EDA

EDA was conducted using Sweetviz and AutoViz to visualize and compare datasets, identify patterns, and understand data distributions.





	Data Type	Missing Values%	Unique Values%	Minimum Value	Maximum Value	DQ Issue
Age	int64	0.000000	2	18.000000	60.000000	No issue
Attrition	object	0.000000	0	nan	nan	No issue
BusinessTravel	object	0.000000	0	nan	nan	No issue
DailyRate	int64	0.000000	60	102.000000	1499.000000	No issue
Department	object	0.000000	0	nan	nan	No issue
DistanceFromHome	int64	0.000000	1	1.000000	29.000000	No issue
Education	int64	0.000000	0	1.000000	5.000000	No issue
EducationField	object	0.000000	0	nan	nan	No issue
EmployeeCount	int64	0.000000	0	1.000000	1.000000	Zero-variance colum: drop before modeling process.
EmployeeNumber	int64	0.000000	100	1.000000	2068.000000	Possible ID colum: drop before modeling process.
EnvironmentSatisfaction	int64	0.000000	0	1.000000	4.000000	No issue
Gender	object	0.000000	0	nan	nan	No issue
HourlyRate	int64	0.000000	4	30.000000	100.000000	No issue
JobInvolvement	int64	0.000000	0	1.000000	4.000000	No issue
JobLevel	int64	0.000000	0	1.000000	5.000000	No issue
JobRole	object	0.000000	0	nan	nan	No issue
JobSatisfaction	int64	0.000000	0	1.000000	4.000000	No issue
MaritalStatus	object	0.000000	0	nan	nan	No issue
MonthlyIncome	int64	0.000000	91	1009.000000	19999.000000	has 114 outliers greater than upper bound (16581.00) or lower than lower bound(-5291.00). Cap them or remove them., has a high correlation with ['JobLevel']. Consider dropping one of them.
MonthlyRate	int64	0.000000	97	2094.000000	26999.000000	No issue
NumCompaniesWorked	int64	0.000000	0	0.000000	9.000000	has 52 outliers greater than upper bound (8.50) or lower than lower bound(-3.50). Cap them or remove them.
Over18	object	0.000000	0	nan	nan	Zero-variance colum: drop before modeling process.
OverTime	object	0.000000	0	nan	nan	No issue
PercentSalaryHike	int64	0.000000	1	11.000000	25.000000	No issue
PerformanceRating	int64	0.000000	0	3.000000	4.000000	has 226 outliers greater than upper bound (3.00) or lower than lower bound(3.00). Cap them or remove them.
RelationshipSatisfaction	int64	0.000000	0	1.000000	4.000000	No issue
StandardHours	int64	0.000000	0	80.000000	80.000000	Zero-variance colum: drop before modeling process.
StockOptionLevel	int64	0.000000	0	0.000000	3.000000	has 85 outliers greater than upper bound (2.50) or lower than lower bound(-1.50). Cap them or remove them.
TotalWorkingYears	int64	0.000000	2	0.000000	40.000000	has 63 outliers greater than upper bound (28.50) or lower than lower bound(-7.50). Cap them or remove them.

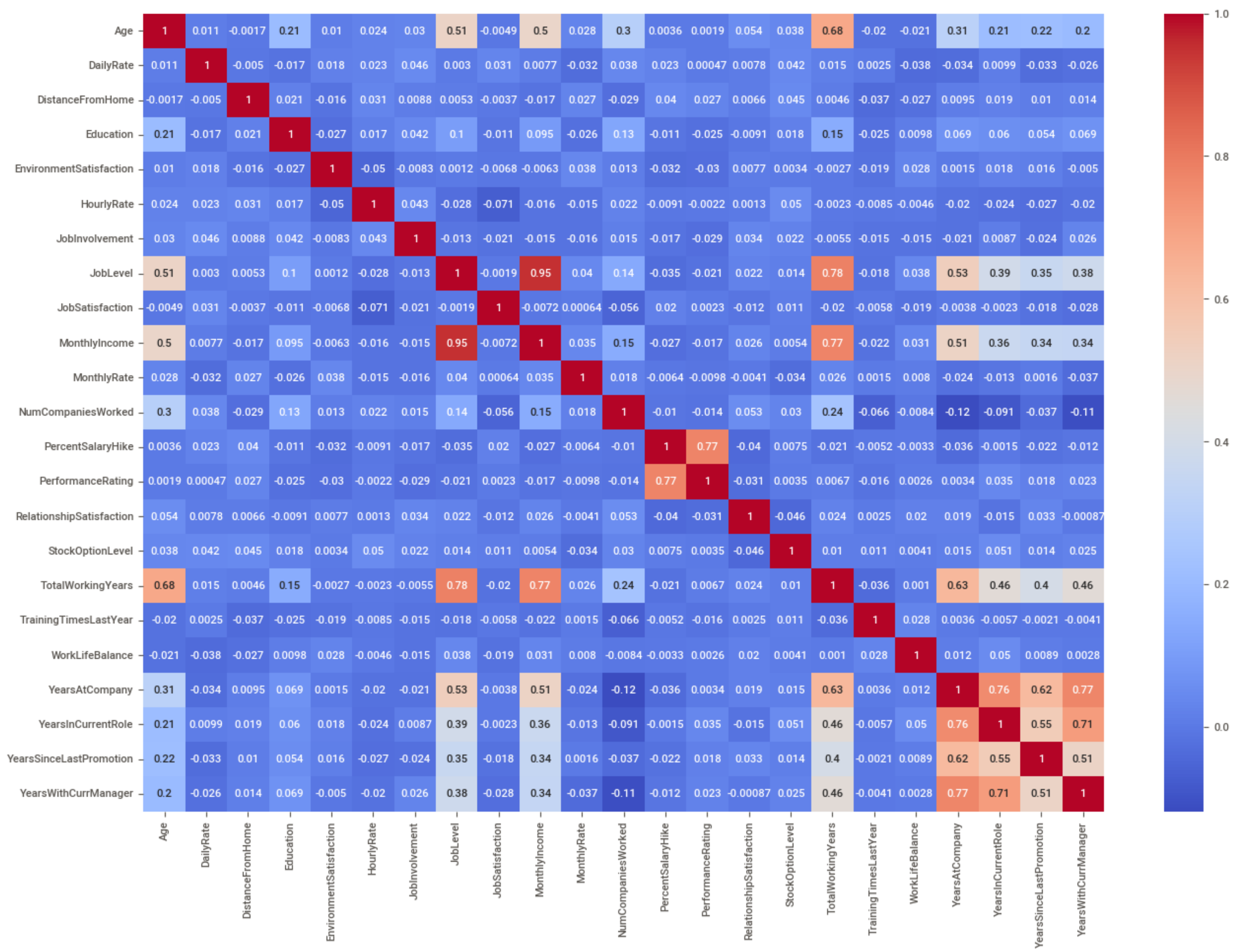
Chapter-3

Feature Selection

3.1 CORRELATION ANALYSIS

Correlation analysis was employed to assess the strength and direction of the relationship between different variables in the dataset, aiding in identifying potential predictors for the target variable and understanding the interdependencies within the data. Correlation analysis provided insights into how variables in the dataset were related to each other, highlighting potential patterns and dependencies that could influence the outcome variable. By examining correlation coefficients, such as Pearson's r, the degree of linear association between pairs of variables was evaluated, guiding feature selection and model development decisions.

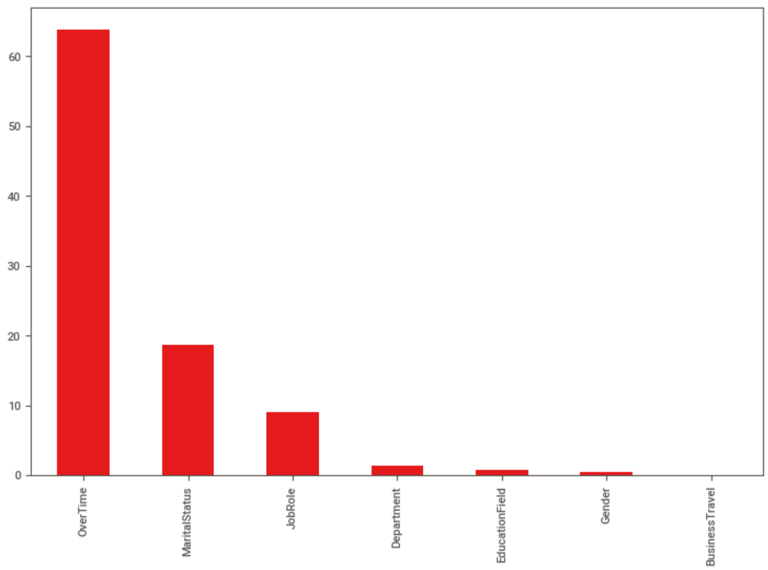
```python
#Correlation Analysis
correlation_matrix = IBM_HR.corr()
plt.figure(figsize=(14,10))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', annot_kws={"size":8})
plt.show()
```



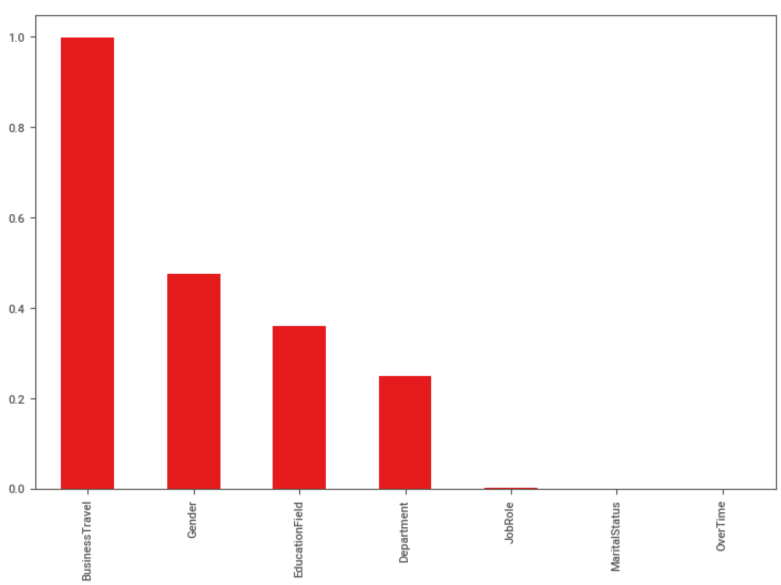
3.2 CHI-SQUARE TEST

Chi-square test assisted in determining the significance of the association between variables, aiding in feature selection for predictive modeling.

```
#Higher the chi value, higher the importance
chi_values = pd.Series(chi_scores[0], index=p.columns)
chi_values.sort_values(ascending=False, inplace=True)
chi_values.plot.bar()
```




```
#If the p value > 0.5, lower the importance
p_values = pd.Series(chi_scores[1], index=p.columns)
p_values.sort_values(ascending=False, inplace=True)
p_values.plot.bar()
```



Chapter 4

Model Implementation

4.1 MODEL SELECTION

In the pursuit of developing an AI tool to address employee performance and attrition, a range of algorithms were meticulously evaluated. The assessment included logistic regression, logistic regression with L1 regularization, random forest, gradient boosting, AdaBoost, support vector machines (SVM), k-nearest neighbors (KNN), XGBoost, and various support vector regressors (SVR) such as linear, polynomial, and radial basis function (RBF) kernels. Additionally, after thorough hyperparameter tuning, further exploration was conducted with algorithms like random forest, gradient boosting, and CatBoost. Each algorithm was scrutinized for its efficacy in predicting and addressing employee behavior and outcomes, ultimately culminating in the selection of logistic regression as the most accurate and interpretable model for the given task.

- Logistic Regression

LOGISTIC REGRESSION

```
# Initialize and train Logistic Regression model
logreg = LogisticRegression()
logreg.fit(X_train_scaled, y_train)
y_pred_logreg = logreg.predict(X_test_scaled)
accuracy_logreg = accuracy_score(y_test, y_pred_logreg)
print("Logistic Regression Accuracy:", accuracy_logreg)
```

```
Logistic Regression Accuracy: 0.8945578231292517
```

- Logistic Regression with L1 Regularization

```
# Apply L1 regularization with Logistic Regression
model = LogisticRegression(penalty='l1', solver='liblinear', random_state=42)
model.fit(X_train_scaled, y_train)

# Predictions
y_pred = model.predict(X_test_scaled)
accuracy_l1_logreg = accuracy_score(y_test, y_pred)
print("Logistic Regression with L1 Regularization Accuracy:", accuracy_l1_logreg)

# Calculate AUC value
y_prob = model.predict_proba(X_test_scaled)[:, 1]
auc_value = roc_auc_score(y_test, y_prob)
print(f'AUC value with L1 Regularization: {auc_value:.2f}')

# Plot ROC curve
fpr, tpr, _ = roc_curve(y_test, y_prob)
plt.figure(figsize=(8, 6))
plt.plot(fpr, tpr, label=f'ROC Curve (AUC = {auc_value:.2f})')
plt.plot([0, 1], [0, 1], linestyle='--', color='gray', label='Random Guess')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC Curve with L1 Regularization')
plt.legend()
plt.show()
```

```
Logistic Regression with L1 Regularization Accuracy: 0.8843537414965986
AUC value with L1 Regularization: 0.77
```

- Random Forest

```
# Initialize and train Random Forest model
rf_classifier = RandomForestClassifier()
rf_classifier.fit(X_train_scaled, y_train)
y_pred_rf = rf_classifier.predict(X_test_scaled)
accuracy_rf = accuracy_score(y_test, y_pred_rf)
print("Random Forest Accuracy:", accuracy_rf)
```

```
Random Forest Accuracy: 0.8775510204081632
```

- Gradient Boosting

GRADIENT BOOSTING

```
# Initialize and train Gradient Boosting model
gb_classifier = GradientBoostingClassifier()
gb_classifier.fit(X_train_scaled, y_train)
y_pred_gb = gb_classifier.predict(X_test_scaled)
accuracy_gb = accuracy_score(y_test, y_pred_gb)
print("Gradient Boosting Accuracy:", accuracy_gb)
```

```
Gradient Boosting Accuracy: 0.891156462585034
```

- AdaBoost

ADABOOST

```
[49] # Initialize and train AdaBoost classifier
adaboost_classifier = AdaBoostClassifier(n_estimators=50, random_state=42)
adaboost_classifier.fit(X_train_scaled, y_train)
y_pred_adaboost = adaboost_classifier.predict(X_test_scaled)
accuracy_adaboost = accuracy_score(y_test, y_pred_adaboost)
print("AdaBoost Accuracy:", accuracy_adaboost)
```

```
AdaBoost Accuracy: 0.8707482993197279
```

◆ SVM

SVM

```
[50] # Support Vector Machine (SVM)
svm_classifier = SVC()
svm_classifier.fit(X_train_scaled, y_train)
y_pred_svm = svm_classifier.predict(X_test_scaled)
accuracy_svm = accuracy_score(y_test, y_pred_svm)
print("SVM Accuracy:", accuracy_svm)
```

```
SVM Accuracy: 0.8877551020408163
```

◆ KNN

KNN

```
# K-Nearest Neighbors (KNN)
knn_classifier = KNeighborsClassifier()
knn_classifier.fit(X_train_scaled, y_train)
y_pred_knn = knn_classifier.predict(X_test_scaled)
accuracy_knn = accuracy_score(y_test, y_pred_knn)
print("KNN Accuracy:", accuracy_knn)
```

```
KNN Accuracy: 0.8605442176870748
```

◆ XGBoost

XGBOOST

```
# Extreme Gradient Boosting (XGBoost)
xgb_classifier = XGBClassifier()
xgb_classifier.fit(X_train_scaled, y_train)
y_pred_xgb = xgb_classifier.predict(X_test_scaled)
accuracy_xgb = accuracy_score(y_test, y_pred_xgb)
print("XGBoost Accuracy:", accuracy_xgb)
```

```
XGBoost Accuracy: 0.8707482993197279
```

- SVR(Linear, Poly, RBF)

```
from sklearn.svm import SVR
from sklearn.metrics import mean_absolute_error, r2_score

# SVR with poly kernel
svr_poly = SVR(kernel='poly')
svr_poly.fit(X_train_scaled, y_train)
y_pred_poly = svr_poly.predict(X_test_scaled)

# SVR with rbf kernel
svr_rbf = SVR(kernel='rbf')
svr_rbf.fit(X_train_scaled, y_train)
y_pred_rbf = svr_rbf.predict(X_test_scaled)

# Evaluate SVR with poly kernel
mae_poly = mean_absolute_error(y_test, y_pred_poly)
r2_poly = r2_score(y_test, y_pred_poly)
print("SVR with Poly Kernel - MAE:", mae_poly)
print("SVR with Poly Kernel - R-squared:", r2_poly)

# Evaluate SVR with rbf kernel
mae_rbf = mean_absolute_error(y_test, y_pred_rbf)
r2_rbf = r2_score(y_test, y_pred_rbf)
print("SVR with RBF Kernel - MAE:", mae_rbf)
print("SVR with RBF Kernel - R-squared:", r2_rbf)
```

```
SVR with Poly Kernel - MAE: 0.24295819208026354
SVR with Poly Kernel - R-squared: -0.1885221647368196
SVR with RBF Kernel - MAE: 0.22449469518424714
SVR with RBF Kernel - R-squared: 0.10624591115817605
```

- Random Forest, Gradient Boosting, CatBoost after Hyperparameter Tuning

CATBOOST

```
# CatBoost Hyperparameter Tuning
catboost_params = {
    'iterations': [50, 100, 200],
    'learning_rate': [0.01, 0.1, 0.2],
    'depth': [4, 6, 8],
    'l2_leaf_reg': [1, 3, 5, 7, 9]
}

catboost_classifier_tuned = GridSearchCV(CatBoostClassifier(random_state=42, verbose=False), catboost_params, cv=5, scoring='accuracy')
catboost_classifier_tuned.fit(X_train_scaled, y_train)

# Display the best hyperparameters
print("Best Hyperparameters for CatBoost:", catboost_classifier_tuned.best_params_)

# Use the best model
y_pred_catboost_tuned = catboost_classifier_tuned.predict(X_test_scaled)
accuracy_catboost_tuned = accuracy_score(y_test, y_pred_catboost_tuned)
print("CatBoost Accuracy after Hyperparameter Tuning:", accuracy_catboost_tuned)
```

```
Best Hyperparameters for CatBoost: {'depth': 6, 'iterations': 200, 'l2_leaf_reg': 5, 'learning_rate': 0.1}
CatBoost Accuracy after Hyperparameter Tuning: 0.891156462585034
```

4.2 MODEL EVALUATION

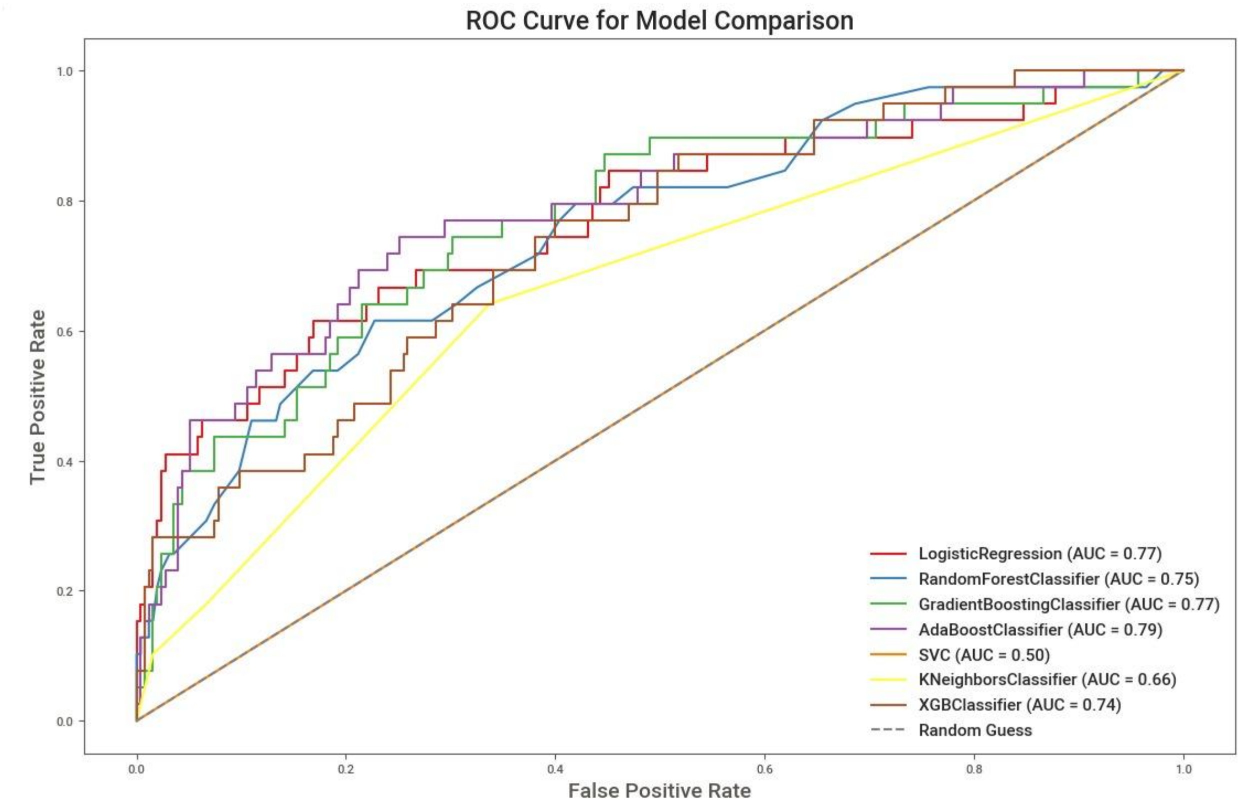
K-fold cross-validation is a widely used technique in machine learning for assessing the performance and generalization ability of a model. The process involves partitioning the dataset into 'k' subsets or folds of approximately equal size. The model is trained 'k' times, each time using a different fold as the testing set and the remaining folds as the training set.

```python
# Define the number of folds for cross-validation
n_folds = 5
kf = KFold(n_splits=n_folds, shuffle=True, random_state=42)

# Initialize models
models = [
    LogisticRegression(penalty='l1', solver='liblinear', random_state=42),
    RandomForestClassifier(),
    GaussianNB(),
    GradientBoostingClassifier(),
    BaggingClassifier(base_estimator=DecisionTreeClassifier(), n_estimators=10, random_state=42),
    AdaBoostClassifier(n_estimators=50, random_state=42)
]
# Lists to store mean accuracies and standard deviations
mean_accuracies = []
std_accuracies = []

# Evaluate models using k-fold cross-validation
for model in models:
    model_name = model.__class__.__name__
    accuracy_scores = cross_val_score(model, X_scaled, y, cv=kf, scoring='accuracy')
    mean_accuracy = accuracy_scores.mean()
    std_accuracy = accuracy_scores.std()

    mean_accuracies.append(mean_accuracy)
    std_accuracies.append(std_accuracy)

    print(f'{model_name} - Mean Accuracy: {mean_accuracy:.2%}, Standard Deviation: {std_accuracy:.4f}')
```

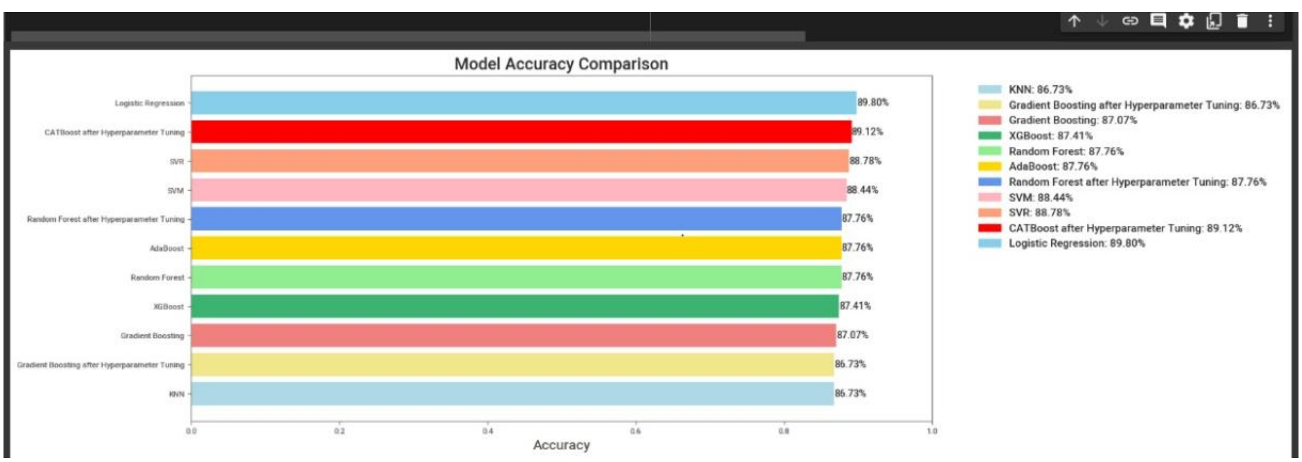
```
LogisticRegression - Mean Accuracy: 85.31%, Standard Deviation: 0.0250
RandomForestClassifier - Mean Accuracy: 84.35%, Standard Deviation: 0.0184
GaussianNB - Mean Accuracy: 75.37%, Standard Deviation: 0.0340
GradientBoostingClassifier - Mean Accuracy: 84.63%, Standard Deviation: 0.0199
BaggingClassifier - Mean Accuracy: 83.95%, Standard Deviation: 0.0167
AdaBoostClassifier - Mean Accuracy: 84.76%, Standard Deviation: 0.0156
```

Model Comparison using ROC Curve:

```python
from sklearn.metrics import roc_curve, roc_auc_score, auc
# Plot ROC curves for each model

plt.figure(figsize=(12, 8))

for model in models:
    model.fit(X_train_scaled, y_train)

    try:
        y_prob = model.predict_proba(X_test_scaled)[:, 1]
    except AttributeError:
        # For models without predict_proba, use decision_function
        y_prob = model.decision_function(X_test)

    fpr, tpr, thresholds = roc_curve(y_test, y_prob)
    roc_auc = auc(fpr, tpr)
    plt.plot(fpr, tpr, label=f'{model.__class__.__name__} (AUC = {roc_auc:.2f})')

plt.plot([0, 1], [0, 1], linestyle='--', color='gray', label='Random Guess')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC Curve for Model Comparison')
plt.legend()
plt.show()
```



Graph on Model Accuracy Comparison:



Chapter 5

Deployment

5.1 DEPLOYMENT

The best-performing models were deployed using a Flask app to predict employee attrition and assist HR in making informed decisions. The AI tool is integrated into a Flask web application, allowing HR and operations teams to access its functionalities through a user-friendly interface. Flask, a lightweight Python web framework, facilitates the development of interactive web applications, making it an ideal choice for deploying AI solutions. The Flask app is hosted on a server, ensuring accessibility to authorized users within the organization. User authentication mechanisms are implemented to secure access to sensitive employee data and ensure compliance with privacy regulations.

https://ibm-attrition-predictor.onrender.com/

Chapter 6

Conclusion and Future work

6.1 CONCLUSION

In conclusion, the development of the AI tool aimed at addressing employee performance and attrition has been a significant milestone in leveraging data-driven insights to enhance organizational effectiveness. By employing machine learning algorithms and predictive analytics, the tool offers HR and operations teams invaluable insights into employee behavior and performance trends, enabling proactive decision-making and targeted interventions. Throughout the project, the primary objectives of reducing turnover rates, retaining key talent, and enhancing employee satisfaction have been diligently pursued. Through the utilization of advanced analytics techniques on the provided dataset, the tool has demonstrated promising capabilities in identifying factors influencing attrition and predicting future employee performance.

6.2 FUTURE WORK

Moving forward, the AI tool can be enhanced by integrating additional data sources beyond Kaggle, refining predictive models, implementing real-time data analysis capabilities, improving the user interface, expanding functionalities to include talent management, incorporating a feedback mechanism for iterative development, and addressing ethical considerations. These advancements aim to provide organizations with a more comprehensive and actionable solution for managing employee performance and attrition, ultimately contributing to improved decision-making, enhanced employee satisfaction, and organizational success.

REFERENCES

Raza, A., Munir, K., Almutairi, M., Younas, F., & Fareed, M. M. S. (2022). Predicting employee attrition using machine learning approaches. Applied Sciences (Basel, Switzerland), 12(13), 6424. doi:10.3390/app12136424

Géron, A. (2019). Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow (2nd ed.). O'Reilly Media.

Rao, A. S., Vardhan, B. V., & Shaik, H. (2021, July 8). Role of exploratory data analysis in data science. 2021 6th International Conference on Communication and Electronics Systems (ICCES). Presented at the 2021 6th International Conference on Communication and Electronics Systems (ICCES), Coimbatre, India. doi:10.1109/icces51350.2021.9488986

Nasteski, V. (2017). An overview of the supervised machine learning methods. Horizons. b, 4(51-62), 56.

Ferreira, A. J., & Figueiredo, M. A. T. (2012). Boosting algorithms: A review of methods, theory, and applications. In Ensemble Machine Learning (pp. 35–85). doi:10.1007/978-1-4419-9326-7_2

Rong, S., & Bao-wen, Z. (2018). The research of regression model in machine learning field. MATEC Web of Conferences, 176, 01033. doi:10.1051/matecconf/201817601033

Kotsiantis, S., Kanellopoulos, D., & Pintelas, P. (01 2006). Data Preprocessing for Supervised Learning. International Journal of Computer Science, 1, 111–117.

Berrar, D. (2019). Cross-Validation. In Encyclopedia of Bioinformatics and Computational Biology (pp. 542–545). doi:10.1016/b978-0-12-809633-8.20349