EMPLOYEE SALARY PREDICTION PROJECT

Aim: The aim of this project is to predict the salary of different employees with relation to multiple factors and features.

Machine learning model consists of:

1- **Data discovery** (importing libraries and dataset)
2- **Data preprocessing**
3- **Exploratory Data Analysis**
4- **Hyperparameter tuning**
5- **Modelling**

1.Data discovery

This process involves the collection and evaluation of data from multiple sources and is used to understand patterns in data. The following steps were taken in order to do so.

- Required libraries were imported

```
Import pandas as pd

Import numpy as np

Import matplotlib.pyplot as plt

Import seaborn as sns
```

- Dataset was read

```
df = pd.read_csv('../input/salary-prediction-
classification/salary.csv') df.head()
```

- Observation and features were measured

```
emp_df = df.copy()

emp_df.shape
```

32561 observations and 15 features

- Names of columns were acquired
  Emp_df.columns

```
Output: Index(['age', 'workclass', 'fnlwgt', 'education', 'education-num',
       'marital-status', 'occupation', 'relationship', 'race', 'sex',
       'capital-gain', 'capital-loss', 'hours-per-week', 'native-country',
       'salary'],
      dtype='object')
```

- Data types of each feature

```
age             int64
workclass       object
fnlwgt          int64
education       object
education-num   int64
marital-status  object
occupation      object
relationship    object
race            object
sex             object
capital-gain    int64
capital-loss    int64
hours-per-week  int64
native-country  object
salary          object
dtype: object
```

- Each feature was looked into using the following code
  Emp_df[‘feature’}.value_counts()
- Missing values (?) were replaced by:
  Emp_df[‘feature’].replace (‘?’ , np.nan, inplace=True)

2. Data preprocessing

Data Preprocessing includes the steps we need to follow to transform or encode data so that it may be easily parsed by the machine.

The main agenda for a model to be accurate and precise in predictions is that the algorithm should be able to easily interpret the data's features.

The majority of the real-world datasets for machine learning are highly susceptible to be missing, inconsistent, and noisy due to their heterogeneous origin.

Applying data mining algorithms on this noisy data would not give quality results as they would fail to identify patterns effectively. Data Processing is, therefore, important to improve the overall data quality.

- Duplicate or missing values may give an incorrect view of the overall statistics of data.
- Outliers and inconsistent data points often tend to disturb the model’s overall learning, leading to false predictions.

Quality decisions must be based on quality data. Data Preprocessing is important to get this quality data, without which it would just be a Garbage In, Garbage Out scenario.

4 Steps in Data Preprocessing

- Data Cleaning
- Data integration
- Data transformation
- Data reduction

In our data set we relied heavily on data cleaning.

- Missing values were replaced with mode
- Duplicates were dropped

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

- Replace ‘?’ by NaN

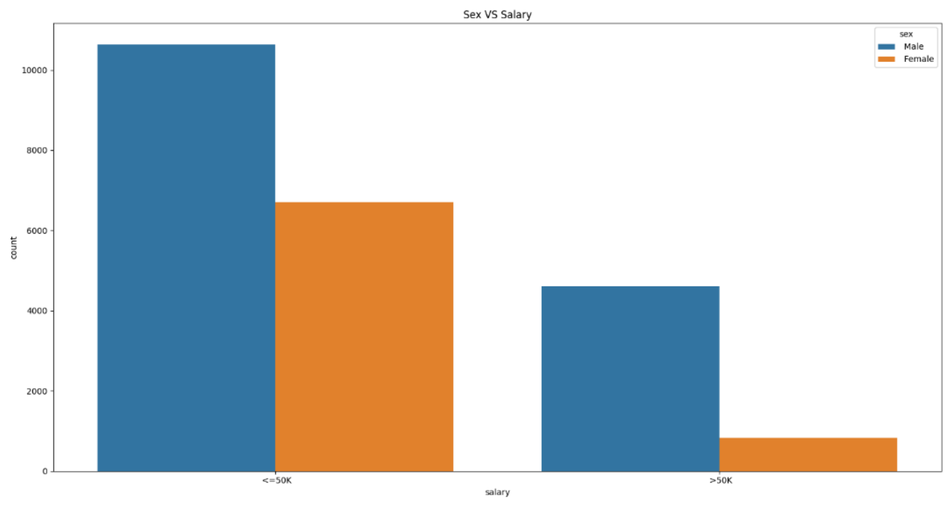
```
Data.replace(‘?’ , np.nan , inplace=True)
```

Exploratory Data Analysis

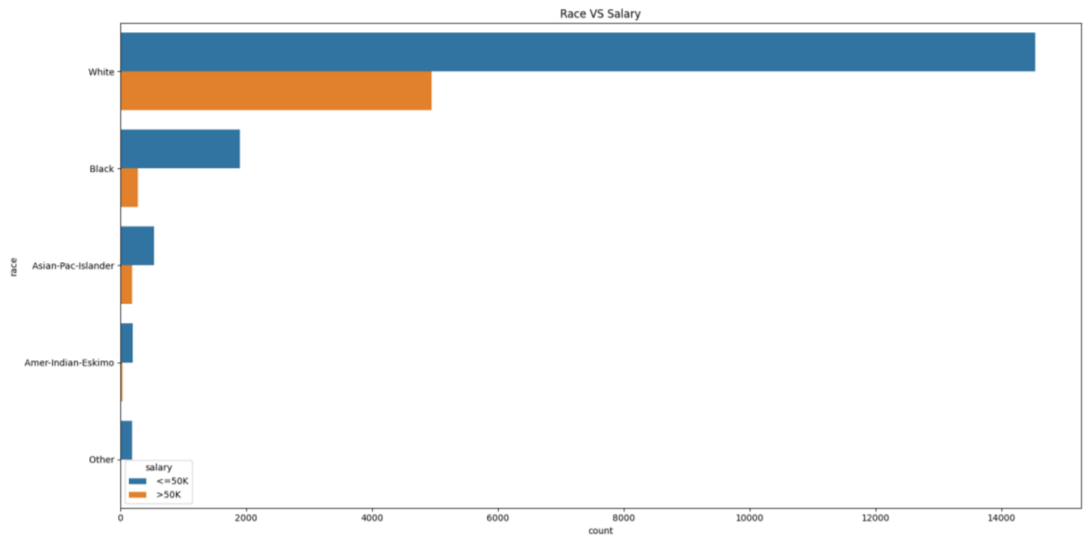
Exploratory data analysis (EDA) is used by data scientists to analyze and investigate data sets and summarize their main characteristics, often employing data visualization methods. It helps determine how best to manipulate data sources to get the answers you need, making it easier for data scientists to discover patterns, spot anomalies, test a hypothesis, or check assumptions.

EDA is primarily used to see what data can reveal beyond the formal modeling or hypothesis testing task and provides a provides a better understanding of data set variables and the relationships between them. It can also help determine if the statistical techniques you are considering for data analysis are appropriate. Originally developed by American mathematician John Tukey in the 1970s, EDA techniques continue to be a widely used method in the data discovery process today. The following were implemented in our project.

-
  ```
  fig = plt.figure(figsize=(10, 6))
  ax = sns.countplot(data=data,x='salary', hue='sex')
  ax.set_title('Sex VS Salary')
  plt.show()
  ```



-
  ```
  plt.figure(figsize=(15, 8))
  ax = sns.countplot(data=data,y='race', hue='salary')
  sns.set_palette('Accent_r')
  ax.set_title('Race VS Salary')
  plt.show()
  ```



The first graph shows the effect of employees’ sex against their salary while the second one shows how does their race affect their salaries.

Training and validation sets

```
x=data.drop(['work-fnl','capital-gain','capital-loss','salary','education-num'],axis=1)
y=data['salary']


x_train , x_test , y_train , y_test = train_test_split(x , y , train_size=0.2 , random_state=0)##80% training & 20% testing###
print('Shape of training feature:', x_train.shape)
print('Shape of testing feature:', x_test.shape)
print('Shape of training label:', y_train.shape)
print('Shape of training label:', y_test.shape)
```

The dataset was divided into 80% training sets and 20% testing sets.

Training sets are used to train a model for predictions while test sets are Used to get an unbiased estimate of the final model performance.

Models used (algorithms)

- **logistic regression**
- **Random forest**
- **Decision tree classifier**

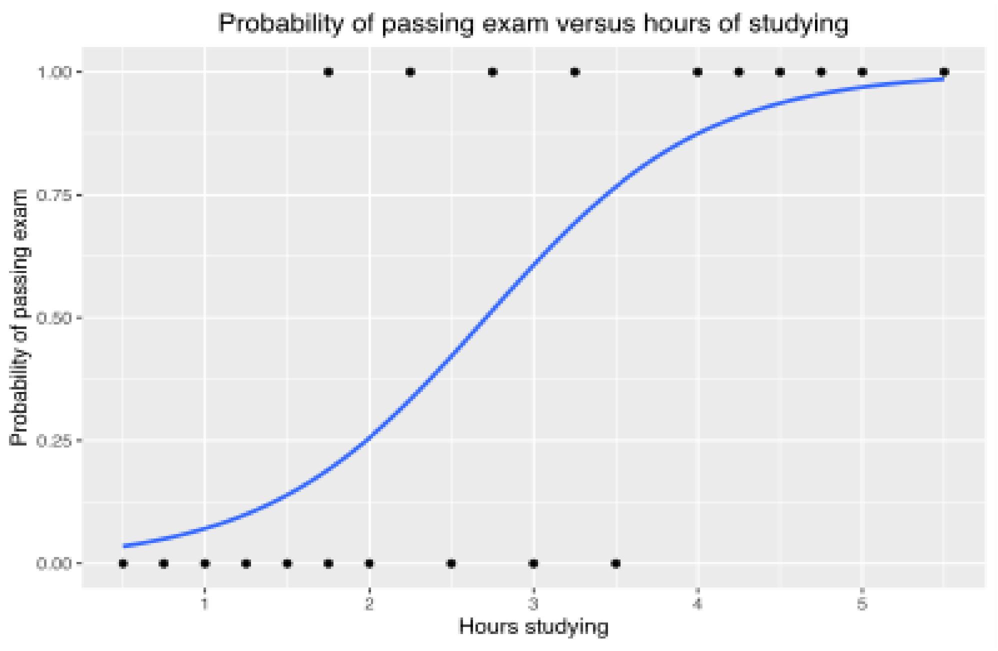
1-logistic regression algorithm

Logistic regression is a supervised learning algorithm used to predict a target variable.

Types of logistic regression:

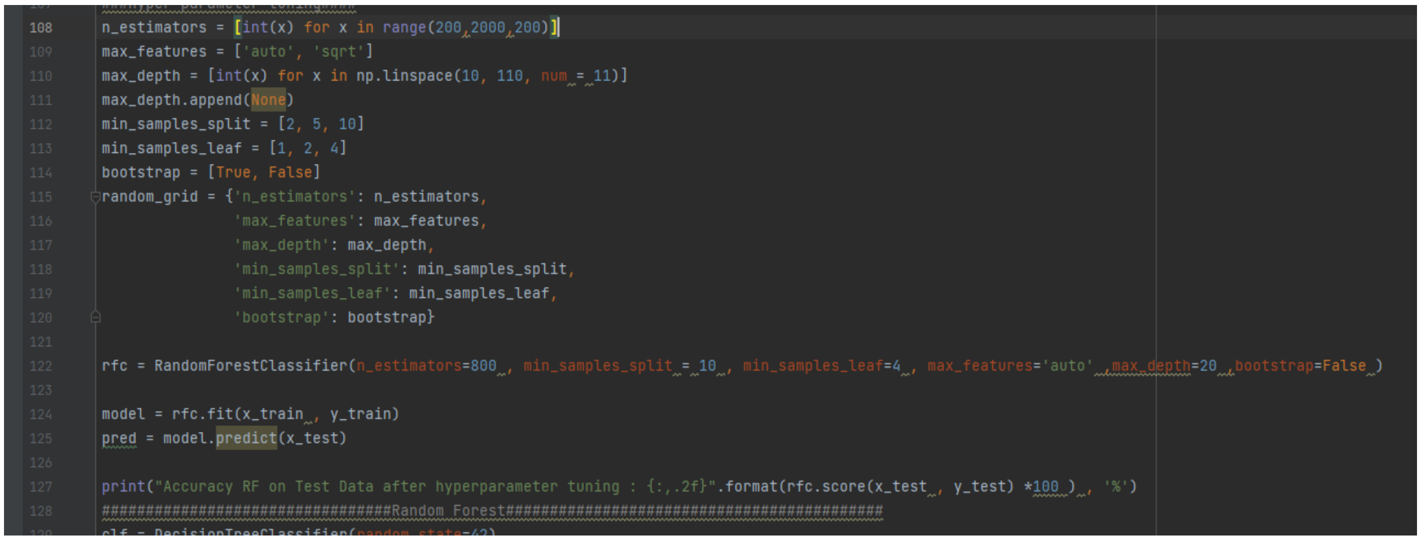
- binary: 2 possible outcomes
- multinomial: more than 2 possible outcomes
- ordinal: ordering of classes is required. Classes do not need to be proportionate. The distance between each class can vary. This was implemented by the following code to predict the employee salaries with the y variable being the salary and x variable being multiple parameters.(work-fnl,capital-gain,capital-loss,education-num.)

-
  ```
  logR=LogisticRegression(random_state=0,
  max_iter=x.shape[0])
  logR.fit(x_train, y_train)
  y_pred = logR.predict(x_test)
  cm = confusion_matrix(y_test, y_pred)
  accuracy = (cm[0,0] + cm[1,1]) / cm.sum()
  print("Accuracy LR: {:,.2f}".format( accuracy * 100),"%")

  accuracy = (cm[0,0] + cm[1,1]) / cm.sum()
  print(cm.sum())
  ```

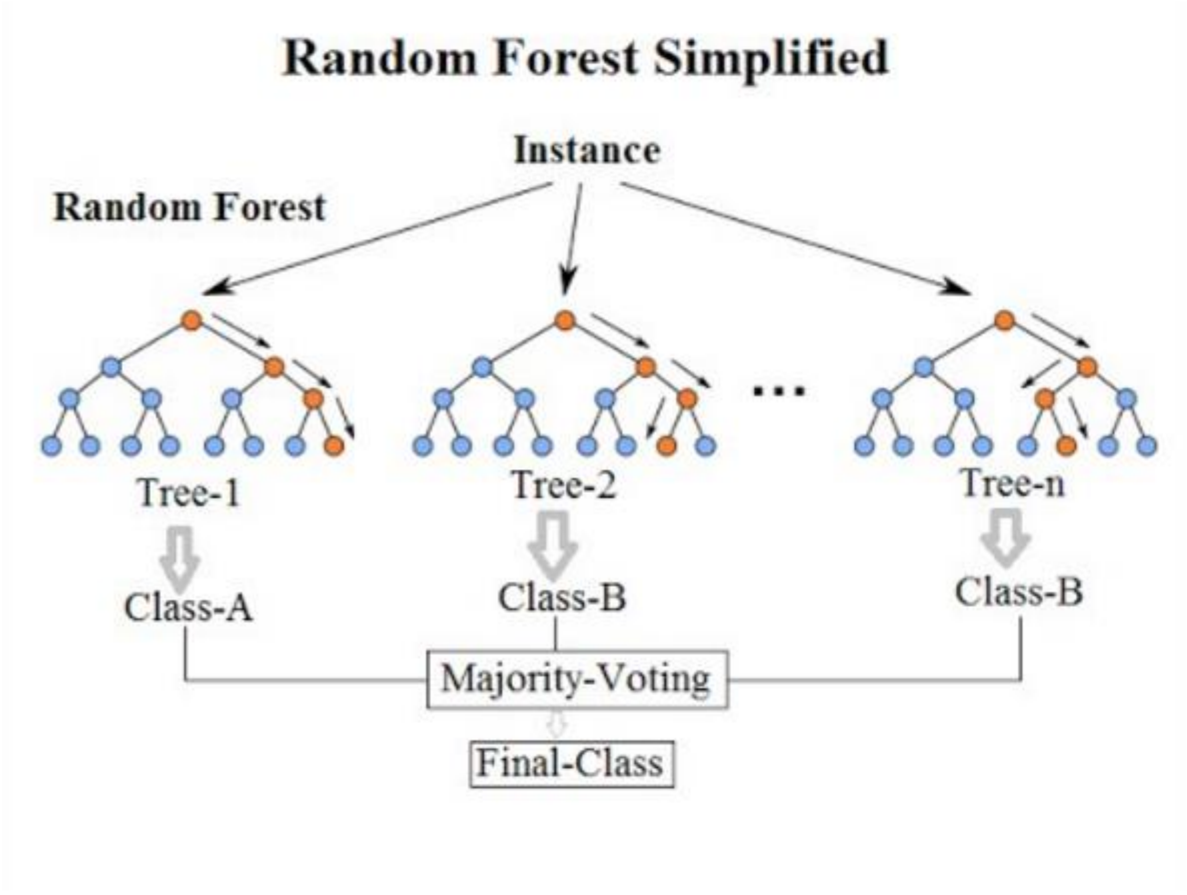


<u>2- Random forest algorithm</u>

This is an algorithm that combines multiple decision trees that operate as an ensemble. Each tree spits out a class prediction and the class with most votes becomes the model's prediction. The predictions made by the individual trees need to have low correlations with each other to ensure optimal results.This was implemented by the following code:

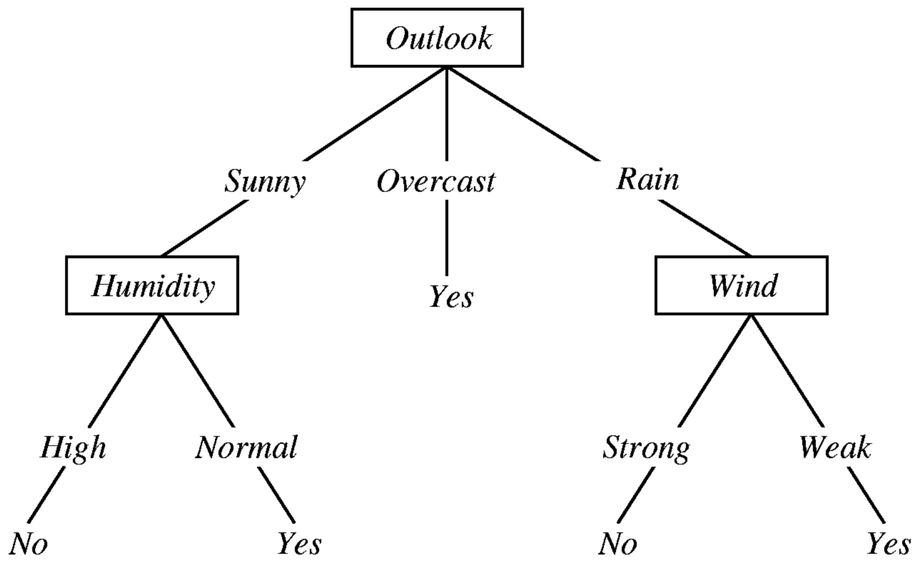


```
n_estimators = [int(x) for x in range(200,2000,200)]
max_features = ['auto', 'sqrt']
max_depth = [int(x) for x in np.linspace(10, 110, num = 11)]
max_depth.append(None)
min_samples_split = [2, 5, 10]
min_samples_leaf = [1, 2, 4]
bootstrap = [True, False]
random_grid = {'n_estimators': n_estimators,
               'max_features': max_features,
               'max_depth': max_depth,
               'min_samples_split': min_samples_split,
               'min_samples_leaf': min_samples_leaf,
               'bootstrap': bootstrap}

rfc = RandomForestClassifier(n_estimators=800 , min_samples_split = 10 , min_samples_leaf=4 , max_features='auto' ,max_depth=20 ,bootstrap=False )

model = rfc.fit(x_train , y_train)
pred = model.predict(x_test)

print("Accuracy RF on Test Data after hyperparameter tuning : {:,.2f}".format(rfc.score(x_test , y_test) *100 ) , '%')
#################################Random Forest###########################################
clf = DecisionTreeClassifier(random_state=42)
```



3- Decision trees:

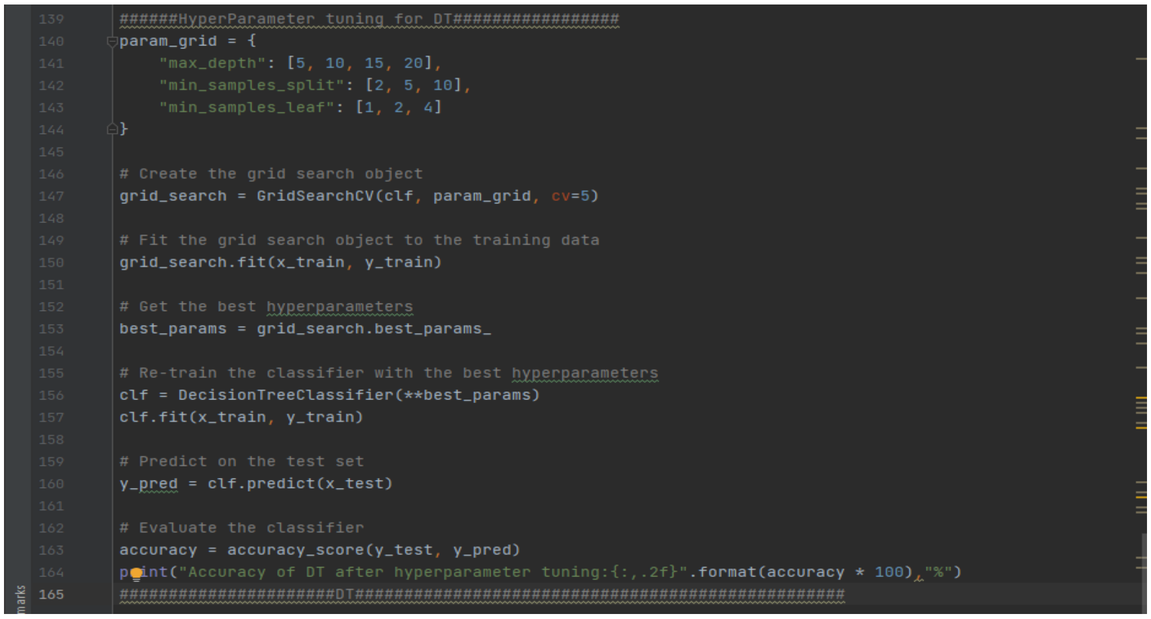
Decision trees are a type of supervised machine learning algorithm where data is continuously split according to a certain parameter. The tree can be explained by two entities known as decision nodes and leaves.



Hyperparameter tuning to improve model results:

Hyperparameter tuning is using top-level parameters for a learning algorithm to optimize results.

In this project hyperparameter tuning was used for the decision tree model.



```python
######HyperParameter tuning for DT#################
param_grid = {
    "max_depth": [5, 10, 15, 20],
    "min_samples_split": [2, 5, 10],
    "min_samples_leaf": [1, 2, 4]
}

# Create the grid search object
grid_search = GridSearchCV(clf, param_grid, cv=5)

# Fit the grid search object to the training data
grid_search.fit(x_train, y_train)

# Get the best hyperparameters
best_params = grid_search.best_params_

# Re-train the classifier with the best hyperparameters
clf = DecisionTreeClassifier(**best_params)
clf.fit(x_train, y_train)

# Predict on the test set
y_pred = clf.predict(x_test)

# Evaluate the classifier
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy of DT after hyperparameter tuning:{:,.2f}".format(accuracy * 100),"%")
######################DT####################################################
```

Conclusion

Machine learning is a powerful and evolving tool to make future predictions from data. This opens the future for very bright opportunities. The speedy rate at which machine learning is developing will result in fast advancements in our world.