Assignment No 1

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from scipy import stats

# Load Dataset
file_path = r"data_engineer_jobs_salaries_2024.csv"
data = pd.read_csv(file_path)

# Display initial rows and information about the dataset
print("First 5 rows of the dataset:")
print(data.head())

print("\nDataset Information:")
print(data.info())

print("\nSummary Statistics:")
print(data.describe())

# Check for missing values
print("\nMissing Values:")
print(data.isnull().sum())

# Drop rows with missing values (if applicable)
data_cleaned = data.dropna()

# Display unique values in categorical columns (if any)
categorical_cols = data_cleaned.select_dtypes(include=['object']).columns
for col in categorical_cols:
    print(f"\nUnique values in '{col}': {data_cleaned[col].unique()}")

# Correlation Matrix (Numerical Data)
```

```
numerical_cols = data_cleaned.select_dtypes(include=['number']).columns
if len(numerical_cols) > 1:
    print("\nCorrelation Matrix:")
    correlation_matrix = data_cleaned[numerical_cols].corr()
    print(correlation_matrix)
    sns.heatmap(correlation_matrix, annot=True, cmap="coolwarm")
    plt.title("Correlation Matrix")
    plt.show()

# Visualization: Pairplot
sns.pairplot(data_cleaned)
plt.title("Pairplot of Numerical Features")
plt.show()

# Example Statistical Analysis
if len(numerical_cols) >= 2:
    print("\nT-Test between the first two numerical columns:")
    col1, col2 = numerical_cols[:2]
    t_stat, p_value = stats.ttest_ind(data_cleaned[col1], data_cleaned[col2])
    print(f"T-Statistic: {t_stat}, P-Value: {p_value}")
```

Assignment No 2

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from pandas.core.common import random_state
from sklearn.linear_model import LinearRegression

df_sal = pd.read_csv('Salary_Data_ For_Assignment2.csv')
df_sal.head()

df_sal.describe()
```

```
plt.title('Salary Distribution Plot')
sns.distplot(df_sal['Salary'])
plt.show()

plt.scatter(df_sal['YearsExperience'], df_sal['Salary'], color = 'lightcoral')
plt.title('Salary vs Experience')
plt.xlabel('Years of Experience')
plt.ylabel('Salary')
plt.box(False)
plt.show()

# Splitting variables
X = df_sal.iloc[:, :1]
y = df_sal.iloc[:, 1:]

# Splitting dataset into test/train
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state = 0)

# Regressor model
regressor = LinearRegression()
regressor.fit(X_train, y_train)

# Prediction result
y_pred_test = regressor.predict(X_test)
y_pred_train = regressor.predict(X_train)

# Prediction on training set
plt.scatter(X_train, y_train, color = 'lightcoral')
plt.plot(X_train, y_pred_train, color = 'firebrick')
plt.title('Salary vs Experience (Training Set)')
plt.xlabel('Years of Experience')
plt.ylabel('Salary')
plt.legend(['X_train/Pred(y_test)', 'X_train/y_train'], title = 'Sal/Exp', loc='best', facecolor='white')
```

```
plt.box(False)
plt.show()

# Prediction on test set
plt.scatter(X_test, y_test, color = 'lightcoral')
plt.plot(X_train, y_pred_train, color = 'firebrick')
plt.title('Salary vs Experience (Test Set)')
plt.xlabel('Years of Experience')
plt.ylabel('Salary')
plt.legend(['X_train/Pred(y_test)', 'X_train/y_train'], title = 'Sal/Exp', loc='best', facecolor='white')
plt.box(False)
plt.show()

print(f'Coefficient: {regressor.coef_}')
print(f'Intercept: {regressor.intercept_}')
```

Assignment No 3

```
import pandas as pd
col_names = ['pregnant', 'glucose', 'bp', 'skin', 'insulin', 'bmi', 'pedigree', 'age', 'label']
pima = pd.read_csv("diabetes_for_Assignment3.csv", header=None, names=col_names, skiprows=1)
pima.head()

feature_cols = ['pregnant', 'insulin', 'bmi', 'age','glucose','bp','pedigree']
X = pima[feature_cols]
y = pima.label

from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, random_state=16)

from sklearn.linear_model import LogisticRegression

logreg = LogisticRegression(random_state=16)
```

```
logreg.fit(X_train, y_train)

y_pred = logreg.predict(X_test)

from sklearn import metrics

cnf_matrix = metrics.confusion_matrix(y_test, y_pred)
cnf_matrix

import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

class_names=[0,1]
fig, ax = plt.subplots()
tick_marks = np.arange(len(class_names))
plt.xticks(tick_marks, class_names)
plt.yticks(tick_marks, class_names)

sns.heatmap(pd.DataFrame(cnf_matrix), annot=True, cmap="YlGnBu" ,fmt='g')
ax.xaxis.set_label_position("top")
plt.tight_layout()
plt.title('Confusion matrix', y=1.1)
plt.ylabel('Actual label')
plt.xlabel('Predicted label')

from sklearn.metrics import classification_report
target_names = ['without diabetes', 'with diabetes']
print(classification_report(y_test, y_pred, target_names=target_names))

y_pred_proba = logreg.predict_proba(X_test)[::,1]
fpr, tpr, _ = metrics.roc_curve(y_test,  y_pred_proba)
auc = metrics.roc_auc_score(y_test, y_pred_proba)
plt.plot(fpr,tpr,label="data 1, auc="+str(auc))
```

```
plt.legend(loc=4)
plt.show()
```

Assignment No 4

```
import pandas as pd
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import train_test_split
from sklearn import metrics
import matplotlib.pyplot as plt
from sklearn.tree import plot_tree

col_names = ['Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickness', 'Insulin', 'BMI', 'DiabetesPedigreeFunction',
'Age', 'Outcome']

pima = pd.read_csv("diabetes_for_Assignment3.csv", header=0, names=col_names)
pima.head()

feature_cols = ['Pregnancies', 'Insulin', 'BMI', 'Age', 'Glucose', 'BloodPressure', 'DiabetesPedigreeFunction']
X = pima[feature_cols]
y = pima['Outcome']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=1)

clf = DecisionTreeClassifier()
clf = clf.fit(X_train, y_train)
y_pred = clf.predict(X_test)

print("Accuracy:", metrics.accuracy_score(y_test, y_pred))

plt.figure(figsize=(12, 8))
plot_tree(clf, feature_names=feature_cols, class_names=['0', '1'], filled=True, rounded=True)
plt.savefig('diabetes.png')
```

```
plt.show()


clf = DecisionTreeClassifier(criterion="entropy", max_depth=3)
clf = clf.fit(X_train, y_train)
y_pred = clf.predict(X_test)

print("Accuracy:", metrics.accuracy_score(y_test, y_pred))

plt.figure(figsize=(12, 8))
plot_tree(clf, feature_names=feature_cols, class_names=['0', '1'], filled=True, rounded=True)
plt.savefig('diabetes.png')
plt.show()
```