AIR QUALITY INDEX ANALYSIS

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from statsmodels.tsa.seasonal import seasonal_decompose
data = {
   "date": ["2024-12-01"] * 24,
   "time": [f"{i:02}:00" for i in range(24)],
   "location": ["CityA"] * 24,
   "PM2.5": [35.6, 37.1, 34.5, 36.2, 33.7, 31.2, 32.5, 30.8, 29.6, 28.7, 27.5, 26.8, 25.6,
24.5, 23.2, 22.1, 21.5, 22.0, 23.5, 24.7, 26.3, 28.5, 30.1, 32.0],
   "PM10": [60.2, 62.3, 58.7, 59.0, 57.5, 55.0, 56.2, 54.5, 52.3, 51.0, 49.5, 48.0, 46.7,
45.5, 44.2, 43.0, 42.7, 43.2, 44.8, 46.0, 48.5, 50.2, 52.7, 54.3],
   "temperature": [22.5, 22.1, 21.9, 21.5,
                  21.2, 21.0, 21.1, 21.0, 22.0, 23.0, 24.2, 25.1, 26.0, 27.0, 28.0, 28.5, 27.5,
26.5, 25.2, 24.0, 23.0, 22.5, 22.1, 22.0],
   "wind_speed": [3.2, 3.0, 3.1, 3.3, 3.4, 3.5, 3.2, 3.1, 3.0, 2.9, 3.1, 3.2, 3.4, 3.5, 3.6,
3.8, 3.7, 3.5, 3.3, 3.2, 3.0, 2.8, 2.7, 2.5],}
df = pd.DataFrame(data)
df['datetime'] = pd.to_datetime(df['date'] + ' ' + df['time'])
df.set_index('datetime', inplace=True)
df_extended = pd.concat([df, df.copy()], ignore_index=False)
df_extended.fillna(df_extended.select_dtypes(include=np.number).mean(),
inplace=True)
plt.figure(figsize=(10, 6))
plt.plot(df_extended.index, df_extended['PM2.5'], label='PM2.5', alpha=0.7)
plt.plot(df_extended.index, df_extended['PM10'], label='PM10', alpha=0.7)
plt.title('Time Series of PM2.5 and PM10')
plt.xlabel('Time')
plt.ylabel('Pollutant Levels')
plt.legend()
plt.show()
```

```
result = seasonal_decompose(df_extended['PM2.5'], model='additive', period=24)
result.plot()

plt.show()
```

SPAM (OR) NOT SPAM

```
import pandas as pd

import numpy as np

from sklearn.model_selection import train_test_split

from sklearn.feature_extraction.text import TfidfVectorizer

from sklearn.naive_bayes import MultinomialNB

from sklearn.metrics import accuracy_score, classification_report, confusion_matrix

data=pd.read_csv('dataset.csv')

print(data.head())

data.dropna(inplace =True)

data['text']=data['text'].str.lower()

X=data['text']

Y=data['label']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

vectorizer = TfidfVectorizer(stop_words='english', max_features=5000)

X_train_tfidf = vectorizer.fit_transform(X_train)

X_test_tfidf = vectorizer.transform(X_test)

model = MultinomialNB()

model.fit(X_train_tfidf, y_train)

y_pred = model.predict(X_test_tfidf)

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

print("\nClassification Report:\n", classification_report(y_test, y_pred))

print("\nConfusion Matrix:\n", confusion_matrix(y_test, y_pred))

new_email = ["Congratulations! You've won a $1000 gift card. Click here to claim."]

new_email_tfidf = vectorizer.transform(new_email)

prediction = model.predict(new_email_tfidf)
```

```
print("\nPrediction for new email:", "Scam" if prediction[0] == 1 else "Not Scam")
```

FUEL AMOUNT PREDICTION

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error
df=pd.read_csv("Fuel_data.csv")
print("Dataset Preview:")
print(df.head())
X = df[['distance']]
y = df['fuel']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
model = LinearRegression()
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
mse = mean_squared_error(y_test, y_pred)
print(f"Mean Squared Error: {mse}")
print("Model Coefficients:")
print(f"Intercept: {model.intercept_}")
print(f"Slope: {model.coef_[0]}")
new_distance = pd.DataFrame({'distance': [150]})
predicted_fuel = model.predict(new_distance)
print(f"Predicted fuel for {new_distance.iloc[0, 0]} km: {predicted_fuel[0]:.2f} liters")
```

HOUSE PRICE PREDICTION

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import Ridge
from sklearn.metrics import mean_squared_error
data = pd.read_csv("house_prices.csv")
X = data[['size', 'bedrooms', 'age']]
```

```
y = data['price']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
model = Ridge(alpha=1.0)
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
print("Mean Squared Error:", mean_squared_error(y_test,y_pred))
```

DIABETES CLASSIFICATION

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, classification_report
data = pd.read_csv("diabetes.csv")
X = data[['age', 'bmi', 'blood_pressure', 'glucose']]
y = data['diabetes']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
model = LogisticRegression()
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
print("Accuracy:", accuracy_score(y_test, y_pred))
print("Classification Report:\n", classification_report(y_test,y_pred))
```

PREDICTIVE ANALYTICS FOR HOSPITALS

DISEASE OUTBREAK PREDICATION

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import PolynomialFeatures
from sklearn.pipeline import make_pipeline
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_absolute_error, r2_score
df = pd.read_csv('disease_outbreak_data.csv')
```

```
df['date'] = pd.to_datetime(df['date'])
df['day_number'] = (df['date'] - df['date'].min()).dt.days
X = df[['day_number']]
y = df['number_of_cases']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)
model = make_pipeline(PolynomialFeatures(degree=2), LinearRegression())
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
mae = mean_absolute_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)
print(f'MAE: {mae:.2f}, R²: {r2:.2f}')
future_days = pd.DataFrame(np.arange(X.max().values[0] + 1,
X.max().values[0] + 31), columns=['day_number'])
future_predictions = model.predict(future_days)
X_test_sorted, y_pred_sorted = zip(*sorted(zip(X_test.values.flatten(),
y_pred)))
plt.scatter(X_test, y_test, color='blue', label='Actual Cases')
plt.scatter(future_days, future_predictions, color='green', marker='x',
label='Future Predictions')
plt.plot(X_test_sorted, y_pred_sorted, color='red', label='Predicted Trend')
plt.xlabel("Days")
plt.ylabel("Number of Cases")
plt.legend()
plt.show()
```

LOAN APPROVAL CLASSIFICATION

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, LabelEncoder
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score, classification_report
Dataset=pd.read_csv('loan_approval_data.csv')
label_encoder = LabelEncoder()
dataset['Employment_Status'] =
label_encoder.fit_transform(dataset['Employment_Status'])
X = dataset.drop('Loan_Status', axis=1)
y = dataset['Loan_Status']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,
random_state=42)
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
model = SVC(kernel='linear')
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
print("Accuracy:", accuracy_score(y_test, y_pred))
print("Classification Report:\n", classification_report(y_test,y_pred))
```

ANIMAL CLASSIFICATION

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.preprocessing import LabelEncoder
from sklearn.metrics import accuracy_score, classification_report
df = pd.read_csv("animal_classification.csv")
df = df.drop(columns=["name"])
le_diet = LabelEncoder()
df["diet"] = le_diet.fit_transform(df["diet"])
le_class = LabelEncoder()
df["class"] = le_class.fit_transform(df["class"])
X=df.drop(columns=["class"])
y = df["class"]
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2,random_state=42)
clf = DecisionTreeClassifier()
clf.fit(X_train, y_train)
y_pred = clf.predict(X_test)
print("Accuracy:", accuracy_score(y_test, y_pred))
print("Classification Report:\n", classification_report(y_test, y_pred,
zero_division=1))
```

EMPLOYEE HOPING PREDICTION

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import classification_report, confusion_matrix,
accuracy_score
df = pd.read_csv("employee_hopping.csv")
df = pd.get_dummies(df, drop_first=True)
df.fillna(df.select_dtypes(include=['number']).mean(), inplace=True)
X = df.drop(columns=["Hopped"])
y = df["Hopped"]
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,
random_state=42)
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
model = RandomForestClassifier(n_estimators=100, random_state=42)
model.fit(X_train_scaled, y_train)
y_pred = model.predict(X_test_scaled)
print(f"Accuracy: {accuracy_score(y_test, y_pred)}")
print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred))
print("Classification Report:\n", classification_report(y_test, y_pred))
```

PATIENT PHYSICAL ACTIVITIES PREDICTION

```
import pandas as pd
import numpy as np
import xgboost as xgb
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, LabelEncoder
from sklearn.metrics import accuracy_score
df = pd.read_csv("mhealth.csv")
label_encoder = LabelEncoder()
df["Activity"] = label_encoder.fit_transform(df["Activity"])
X = df.drop(columns=["Activity"])
y = df["Activity"]
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
model = xgb.XGBClassifier(n_estimators=100, learning_rate=0.1,
random_state=42)
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print(f"Model Accuracy: {accuracy:.2f}")
y_pred_original = label_encoder.inverse_transform(y_pred)
print("Predicted Activities:", y_pred_original)
```

SHOPPING MALL CUSTOMER SEGMENTATION

```
import pandas as pd
import numpy as np
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler
df = pd.read_csv("shoppingmall_customer_segmentation.csv")
X = df[['Age', 'Annual_Income', 'Spending_Score', 'Gender',
'Membership_Duration']]
X = pd.get_dummies(X, columns=['Gender'], drop_first=True)
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
wcss = []
for i in range(1, 11):
   kmeans = KMeans(n_clusters=i, init='k-means++', random_state=42)
   kmeans.fit(X_scaled)
```

```
    wcss.append(kmeans.inertia_)
optimal_k = 5
kmeans = KMeans(n_clusters=optimal_k, init='k-means++', random_state=42)
df['Cluster'] = kmeans.fit_predict(X_scaled)
df.to_csv("shopping_mall_customers_clustered.csv", index=False)
print("Clustering complete. File saved.")
```

CUSTOMER SEGMENTATION

```
import pandas as pd
import numpy as np
from sklearn.preprocessing import StandardScaler
from sklearn.cluster import KMeans
df = pd.read_csv("customer_segmentation.csv")
features = df[['Age', 'Annual_Income', 'Spending_Score']]
scaler = StandardScaler()
X_scaled = scaler.fit_transform(features)
num_clusters = 4
kmeans = KMeans(n_clusters=num_clusters, random_state=42, n_init=10)
df['Cluster'] = kmeans.fit_predict(X_scaled)
print("Customer segmentation completed and saved as
'customer_segmented_dataset.csv'.")
```

CLASSIFY IRIS FLOWERS INTO SPECIES (SETOSA, VERSICOLOR OR VIRGINCIA) BASED ON THEIR SEPAL & PETAL MEASUREMENT

```
import pandas as pd

from sklearn.model_selection import train_test_split

from sklearn.ensemble import RandomForestClassifier

from sklearn.preprocessing import StandardScaler

from sklearn.metrics import accuracy_score, classification_report,
confusion_matrix

df = pd.read_csv("iris.csv")

X = df[['sepal_length', 'sepal_width', 'petal_length', 'petal_width']]

y = df['species']

X_scaled = scaler.fit_transform(X)

X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.3,
random_state=42)

model = RandomForestClassifier(n_estimators=100, random_state=42)

model.fit(X_train, y_train)
```

```
y_pred = model.predict(X_test)
print(f"Accuracy: {accuracy_score(y_test, y_pred)}")
print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred))
print("Classification Report:\n", classification_report(y_test, y_pred))
predictions_df = pd.DataFrame({"Actual": y_test, "Predicted": y_pred})
print("\nPredicted Species for Test Data:\n")
print(predictions_df.to_string(index=False))
```