# Name: Maaz Ahmad |

Reg B23F0722AI170 |
AI Yellow |
Abdullah Sajid
Date 06-12-2025
Machine Learning Lab 12, Colab style notebook
Each task from the provided lab is implemented in a separate cell.

---

## Lab No # 12

## Task no 1

```
# Install required packages, run this cell in Colab if needed
try:
    import imblearn
except Exception:
    import sys
    !{sys.executable} -m pip install imbalanced-learn --quiet
```

## Description:

**Installation of Libraries:** "Is step mein hum imbalanced-learn library install kar rahe hain agar wo environment mein majood nahi hai. Yeh library SMOTE use karne ke liye zaroori hai taake hum data imbalance ko handle kar sakein."

---

## Lab Task 2

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split, StratifiedKFold, RandomizedSearchCV, cross_val_score
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, confusion_matrix, roc_auc_score, roc_curve
import joblib
from imblearn.over_sampling import SMOTE

# Load Titanic dataset from seaborn
df = sns.load_dataset('titanic')
print('Dataset shape', df.shape)
print(df.head())
```

## Description:

**Importing Libraries and Loading Dataset:** "Yahan hum ne tamam zaroori libraries import ki hain (pandas, numpy, seaborn, sklearn etc.). Iske baad hum ne Seaborn se 'Titanic' dataset load kiya hai aur uska shape aur pehli 5 rows print ki hain taake data ka structure dekh sakein.

## Task no 3

```
print('Missing value counts')
print(df.isnull().sum())

if 'deck' in df.columns:
    df = df.drop(columns=['deck'])

df['age'] = df['age'].fillna(df['age'].median())
df['fare'] = df['fare'].fillna(df['fare'].median())

df['embarked'] = df['embarked'].fillna(df['embarked'].mode()[0])

if 'who' not in df.columns and 'cabin' in df.columns:
    df['has_cabin'] = df['cabin'].notnull().astype(int)

print('After imputation missing counts')
print(df.isnull().sum())
```

## Description:

**Data Cleaning and Missing Value Imputation:** "Is cell mein hum missing values ko handle kar rahe hain.

Deck column mein bohot zyada missing values thin isliye usay drop kar diya.

Age aur Fare ki missing values ko 'Median' se fill kiya.

Embarked ko 'Mode' (sab se common value) se fill kiya. End mein hum check kar rahe hain ke ab koi null value baqi to nahi."

---

## Task no 4

```
Q1 = df['fare'].quantile(0.25)
Q3 = df['fare'].quantile(0.75)
IQR = Q3 - Q1
lower = Q1 - 1.5 * IQR
upper = Q3 + 1.5 * IQR

df['fare_clipped'] = df['fare'].clip(lower, upper)
print('Fare clipped summary')
print(df['fare_clipped'].describe())
```

## Description:

**Outlier Handling (Clipping):** "Data mein outliers (bohot bari values) ko control karne ke liye hum IQR method use kar rahe hain. Hum ne Fare column ki values ko lower aur upper bound ke darmiyan clip kar diya hai taake model par bura asar na paray."

---

## Task no 5

```
if 'sibsp' in df.columns and 'parch' in df.columns:
    df['family_size'] = df['sibsp'] + df['parch'] + 1
    df['is_alone'] = (df['family_size'] == 1).astype(int)

selected_features = ['pclass', 'sex', 'age', 'fare_clipped', 'embarked', 'family_size', 'is_alone']
print(df[selected_features].head())
```

## Description:

**Feature Engineering** "Yahan hum existing columns se naye features bana rahe hain:

**Family Size:** SibSp aur Parch ko add kar ke banaya.

***Is Alone:*** Agar family size 1 hai to banda akela hai (1), warna nahi (0). Yeh naye features model ko better prediction mein help karenge."

---

## Task no 6

```
features = ['pclass', 'sex', 'age', 'fare_clipped', 'embarked', 'family_size', 'is_alone']
X = df[features].copy()
y = df['survived'].astype(int)

X = pd.get_dummies(X, columns=['sex', 'embarked', 'pclass'], drop_first=True)

numeric_cols = ['age', 'fare_clipped', 'family_size']
scaler = StandardScaler()
X[numeric_cols] = scaler.fit_transform(X[numeric_cols])

print('Feature matrix shape', X.shape)
print(X.head())
```

## Description:

**Data Encoding and Scaling** "Machine learning models sirf numbers samajhte hain.

**Encoding:** Hum ne Sex, Embarked, aur Pclass ko One-Hot Encoding (dummies) ke zariye numbers mein convert kiya.

**Scaling:** Age, Fare, aur Family Size ko StandardScaler use kar ke scale kiya taake tamam features ki range same ho jaye."

---

## Task no 7

```
pca = PCA(n_components=2)
X_pca = pca.fit_transform(X)
print('Explained variance ratio', pca.explained_variance_ratio_)

plt.figure()
plt.plot(np.cumsum(pca.explained_variance_ratio_))
plt.xlabel('components')
plt.ylabel('cumulative explained variance')
plt.title('PCA explained variance')
plt.show()
```

## Description:

**Dimensionality Reduction (PCA)** "Hum PCA (Principal Component Analysis) use kar ke dekh rahe hain ke data ka variance kitne components explain kar rahe hain. Yeh hamain batata hai ke kya hum features kam kar ke bhi data ki information retain kar sakte hain ya nahi."

---

## Task no 8

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, stratify=y, random_state=42)
print('Train shape', X_train.shape, 'Test shape', X_test.shape)
```

## Description:

**Train-Test Split** "Model ko train aur evaluate karne ke liye hum data ko do hisson mein divide kar rahe hain:

**Training Set (80%):** Model ko sikhane ke liye.

**Testing Set (20%):** Model ki performance check karne ke liye. Stratify=y use kiya hai taake dono sets mein survived/not-survived ki ratio same rahe."

---

## Task no 9

```
print('Class distribution before', y_train.value_counts().to_dict())
smote = SMOTE(random_state=42)
X_train_res, y_train_res = smote.fit_resample(X_train, y_train)
print('Class distribution after', pd.Series(y_train_res).value_counts().to_dict())
```

## Description:

**Handling Class Imbalance (SMOTE)** "Titanic dataset mein 'Not Survived' ki tadaad 'Survived' se zyada hai. Is imbalance ko theek karne ke liye hum SMOTE (Synthetic Minority Over-sampling Technique) use kar rahe hain, jo minority class (Survived) ke artificial samples bana kar classes ko barabar kar deta hai."

---

## Task no 10

```
model = LogisticRegression(solver='saga', penalty='elasticnet', l1_ratio=0.5, max_iter=5000, random_state=42)
model.fit(X_train_res, y_train_res)

coef = pd.Series(model.coef_.ravel(), index=X.columns)
print('Top positive coefficients')
print(coef.sort_values(ascending=False).head())
print('Top negative coefficients')
print(coef.sort_values().head())
```

## Description:

**Model Training (Logistic Regression)** "Ab hum LogisticRegression model ko train kar rahe hain. Hum ne ElasticNet penalty use ki hai taake overfitting se bacha ja sake. Training ke baad hum ne check kiya ke kaunse features (coefficients) survival ke liye sab se zyada important hain (Positive coeff matlab survival chance zyada, Negative matlab kam)."

---

## Task no 11

```
skf = StratifiedKFold(n_splits=5, shuffle=True, random_state=42)
scores = cross_val_score(model, X_train_res, y_train_res, cv=skf, scoring='f1')
print('Cross val F1 mean', scores.mean(), 'std', scores.std())
```

## Description:

**Cross Validation:** "Sirf ek train-test split kafi nahi hota, isliye hum Stratified K-Fold Cross Validation use kar rahe hain. Yeh data ko 5 hisson mein tod kar model ko 5 dafa train aur test karta hai taake hamain model ki average performance (F1 Score) ka pata chale."

---

## Task no 12

```
param_dist = {
    'C': [0.01, 0.1, 1, 10, 100],
    'penalty': ['l1', 'l2'],
    'l1_ratio': [0, 0.5, 1]
}

search = RandomizedSearchCV(
    LogisticRegression(solver='saga', max_iter=5000, random_state=42),
    param_distributions=param_dist,
    n_iter=10,
    cv=3,
    scoring='f1',
    random_state=42,
    n_jobs=1
)
search.fit(X_train_res, y_train_res)
print('Best params', search.best_params_)
best_model = search.best_estimator_
```

## Description:

**Hyperparameter Tuning:** "Model ki performance ko mazeed behtar karne ke liye hum RandomizedSearchCV use kar rahe hain. Yeh C, penalty, aur l1_ratio ki mukhtalif combinations ko test karega aur hamain best parameters dhoond kar dega."

---

## Task no 13

```
y_pred = best_model.predict(X_test)
y_proba = best_model.predict_proba(X_test)[:, 1]

print('Accuracy', accuracy_score(y_test, y_pred))
print('Precision', precision_score(y_test, y_pred))
print('Recall', recall_score(y_test, y_pred))
print('F1', f1_score(y_test, y_pred))
print('ROC AUC', roc_auc_score(y_test, y_proba))
print('Confusion matrix')
print(confusion_matrix(y_test, y_pred))

fpr, tpr, _ = roc_curve(y_test, y_proba)
plt.figure()
plt.plot(fpr, tpr)
plt.xlabel('False positive rate')
plt.ylabel('True positive rate')
plt.title('ROC curve')
plt.show()
```

## Description:

**Model Evaluation and Metrics:** "Best model ko use karte hue hum ne Test set par predictions ki hain. Hum ne yeh metrics calculate kiye:

**Accuracy, Precision, Recall, F1 Score:** Model ki quality check karne ke liye.

**Confusion Matrix:** Dekhne ke liye ke kitne True Positives aur False Negatives hain.

**ROC Curve:** Model ki discrimination power dekhne ke liye graph plot kiya."

---

## Task no 14

```
joblib.dump(best_model, 'titanic_logreg_model.joblib')
joblib.dump(scaler, 'titanic_scaler.joblib')
print('Model and scaler saved to disk')
```

## Description:

**Saving the Model:** "Aakhir mein hum ne apne train kiye hue model aur scaler ko joblib ke zariye save kar liya hai. Is file ko baad mein kisi application ya website mein use kiya ja sakta hai bina dobara training kiye."

---

## Lab Summary:

**Objective:** Is lab mein hum ne Titanic dataset par complete Machine Learning pipeline implement ki.

**Data Cleaning:** Sab se pehle hum ne missing values ko Median aur Mode se fill kiya aur outliers ko handle kiya.

**Feature Engineering: ** Data ko mazeed behtar banane ke liye 'Family Size' aur 'Is Alone' ke naye columns create kiye.

**Preprocessing:** Categorical variables (jaise Sex, Embarked) ki One-Hot Encoding ki aur Numerical variables ki Scaling ki taake model unhein sahi samajh sake.

**Handling Imbalance:** Kyun ke dataset mein classes barabar nahi thin, isliye hum ne SMOTE technique use kar ke data ko balance kiya.

**Model Tuning:** Logistic Regression model use kiya aur RandomizedSearchCV ke zariye best hyperparameters find kiye taake best results milen.

**Evaluation:** Model ki performance ko check karne ke liye Accuracy, F1-Score, Confusion Matrix aur ROC Curve ka analysis kiya.

**Deployment:** Aakhir mein train kiye hue model aur scaler ko save kar liya gaya taake future mein use kiya ja sake.

---