MAINCRAFTS TECHNOLOGY
www.maincrafts.com
hr@maincrafts.com

📊 Data Science with Python Internship – Task 5

📌 Task Description

In Task-4 you focused on visual insights and storytelling with charts.
Now, take the next step: turn those insights into a **predictive model**.
In **Task-5**, you will build a **complete machine learning pipeline** for the **Titanic dataset** that:

- Cleans and preprocesses data (impute missing values, encode categoricals, scale numerics)
- Trains a **Logistic Regression** classifier (a classic baseline for binary classification)
- Evaluates with **accuracy, precision/recall/F1, confusion matrix, ROC-AUC**
- Uses **k-fold cross-validation** for more reliable performance estimates
- **Persists** the trained pipeline to disk (so it can be reused later)
- Produces a **short, clear report** summarizing what works and why

This mirrors the **minimum professional workflow** in DS/ML projects.

✅ What You Will Build (Requirements)

Data Prep

- Target: Survived
- Features (suggested): Pclass, Sex, Age, SibSp, Parch, Fare, Embarked
- Missing values:
  - Numeric (Age, Fare) → median
  - Categorical (Embarked) → most frequent
- Encoding: One-Hot for categoricals (Sex, Embarked, optionally Pclass)
- Scaling: Standardize numeric features (helps linear models)

Modeling & Evaluation

- Split train/test (e.g., 80/20, stratified by target)
- Build a **scikit-learn Pipeline** with a **ColumnTransformer**
- Fit **LogisticRegression (max_iter=1000)**

- Metrics on test set:
  - **classification_report** (precision, recall, F1)
  - **Confusion Matrix** plot
  - **ROC Curve** + **ROC-AUC**
- **Cross-validation**: 5-fold ROC-AUC on full dataset (report mean ± std)

Model Persistence

- Save fitted pipeline to model.joblib
- Demonstrate reloading and predicting on a few rows

Documentation

- In Markdown cells: brief notes on choices (features, imputation, encoding), and interpretation of metrics
- 5-10 bullet takeaways: which features matter conceptually, strengths/limits of the baseline, next steps

---

✅ **Deliverable**

- **Jupyter Notebook (.ipynb)** that:
  1. loads/cleans data → 2) builds pipeline → 3) trains/evaluates → 4) cross-validates → 5) saves model → 6) concise report
- **Saved model file**: model.joblib
- Optional: an images/ folder containing saved plots

✅ **Why This Task Is Important**

Skill	Why it matters
Reproducible Pipelines	Clean, auditable steps you can rerun anytime
Proper Evaluation	Avoids overfitting & misleading metrics
Baseline Modeling	Logistic Regression is a transparent, explainable baseline
Model Persistence	Real projects deploy or reuse trained models
Communication	Short, clear reporting is critical in DS jobs

✅ Sample Starter Code (Concise & Expandable)

python

```python
# 1) Imports
import pandas as pd, numpy as np
from sklearn.model_selection import train_test_split, cross_val_score
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
from sklearn.impute import SimpleImputer
from sklearn.preprocessing import OneHotEncoder, StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import (classification_report, roc_auc_score,
                             ConfusionMatrixDisplay, RocCurveDisplay)
import joblib, matplotlib.pyplot as plt

# 2) Load data
df = pd.read_csv("titanic.csv")  # ensure file available (Kaggle Titanic)

# 3) Select features/target
X = df[["Pclass","Sex","Age","SibSp","Parch","Fare","Embarked"]]
y = df["Survived"].astype(int)

# 4) Define feature groups
num_cols = ["Age","SibSp","Parch","Fare"]
cat_cols = ["Pclass","Sex","Embarked"]  # treat Pclass as categorical for OHE

# 5) Preprocessor: impute + scale/encode
preprocess = ColumnTransformer([
    ("num", Pipeline([
        ("imp", SimpleImputer(strategy="median")),
        ("scaler", StandardScaler())
    ]), num_cols),
    ("cat", Pipeline([
        ("imp", SimpleImputer(strategy="most_frequent")),
        ("ohe", OneHotEncoder(handle_unknown="ignore"))
    ]), cat_cols)
])
```

```python
# 6) Full pipeline: preprocess + model
pipe = Pipeline([
    ("prep", preprocess),
    ("clf", LogisticRegression(max_iter=1000))
])

# 7) Train/test split (stratify for class balance)
X_tr, X_te, y_tr, y_te = train_test_split(
    X, y, test_size=0.2, random_state=42, stratify=y
)

# 8) Fit and evaluate
pipe.fit(X_tr, y_tr)

y_pred = pipe.predict(X_te)
y_proba = pipe.predict_proba(X_te)[:, 1]

print("Classification report:\n", classification_report(y_te, y_pred))
print("ROC-AUC:", roc_auc_score(y_te, y_proba))

ConfusionMatrixDisplay.from_predictions(y_te, y_pred)
plt.title("Confusion Matrix (Test)")
plt.show()

RocCurveDisplay.from_predictions(y_te, y_proba)
plt.title("ROC Curve (Test)")
plt.show()
```

```python
# 9) Cross-validation (5-fold ROC-AUC)
cv_auc = cross_val_score(pipe, X, y, cv=5, scoring="roc_auc")
print(f"CV ROC-AUC: {cv_auc.mean():.3f} ± {cv_auc.std():.3f}")

# 10) Persist and reload
joblib.dump(pipe, "model.joblib")
model = joblib.load("model.joblib")
print("Sample predictions:", model.predict(X_te.head(5)))
```

✅ Helpful Free Resources (Expanded)

Topic	Link
Scikit-learn “Getting Started”	https://scikit-learn.org/stable/getting_started.html
Pipelines & ColumnTransformer	https://scikit-learn.org/stable/modules/compose.html
Logistic Regression (sklearn)	https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
Model Evaluation (metrics)	https://scikit-learn.org/stable/modules/model_evaluation.html
Cross-Validation	https://scikit-learn.org/stable/modules/cross_validation.html
Titanic Dataset (Kaggle)	https://www.kaggle.com/c/titanic/data
Data Cleaning in Pandas	https://pandas.pydata.org/docs/user_guide/index.html

✅ Extra Improvement Ideas (Optional but Recommended)

Enhancement	Why it helps
Feature Engineering	Try FamilySize = SibSp + Parch; IsAlone flag; Title from Name
Hyperparameter Tuning	GridSearchCV / RandomizedSearchCV for C, penalty, class_weight
Class Imbalance Handling	Use class_weight="balanced" if classes are skewed
Model Comparison	Compare Logistic Regression vs. RandomForest baseline
Calibration	Check probability calibration (CalibratedClassifierCV)
Explainability	Feature importance (for linear models via coefficients)

✅ Tools You Can Use

- **Google Colab** or **Jupyter Notebook** locally
- **Python Libraries**: pandas, numpy, scikit-learn, matplotlib, joblib
- **Version Control**: Git/GitHub (recommended)