Importing Libraries

```
from imblearn.over_sampling import RandomOverSampler
import pandas as pd
import math
import joblib
import numpy as np
from sklearn.preprocessing import MinMaxScaler
from sklearn.feature_selection import SelectKBest, chi2
from sklearn.datasets import make_classification
from sklearn.ensemble import RandomForestClassifier
from matplotlib import pyplot as plt
from sklearn.model_selection import train_test_split
from imblearn.over_sampling import SMOTE
from imblearn.over_sampling import ADASYN
from imblearn.over_sampling import RandomOverSampler
from lightgbm import LGBMClassifier
from sklearn.model_selection import train_test_split
from sklearn import metrics
from sklearn.metrics import accuracy_score , precision_score, recall_score, f1_score, confusion_matrix
import warnings
warnings.filterwarnings('ignore')
from sklearn.preprocessing import LabelEncoder
```

Uploading File

```
from google.colab import files
uploaded = files.upload()
```

Choose files No file chosen Upload widget is only available when the cell has been executed in the current browser session.
Please rerun this cell to enable.

```
Saving 0_liver.csv to 0_liver.csv
```

```
df = pd.read_csv('0_liver.csv')
df.head()
```

	age	gender	bmi	alcohol_consumption	smoking_status	hepatitis_b	hepatitis_c	liver_function_score	alpha
0	68	Female	18.1	Regular	Former	0	0	51.9	
1	81	Female	19.9	Occasional	Never	0	0	41.6	
2	58	Female	25.5	Never	Never	0	0	76.0	
3	44	Male	16.0	Never	Former	0	0	50.3	
4	72	Male	21.0	Occasional	Former	0	0	39.5	

Preprocessing

```
#Separating Outcome column
y = df['liver_cancer']
#label encoding
le = LabelEncoder()
# List of categorical columns to encode
cols = ['gender', 'alcohol_consumption', 'smoking_status', 'physical_activity_level']
# Apply label encoding and directly update the DataFrame
for col in cols:
    df[col] = le.fit_transform(df[col])
```

```
# Save the encoder as a .pkl file
joblib.dump(le, "label_encoder.pkl")
# Download the file in Colab
from google.colab import files
files.download("label_encoder.pkl")
```

```
df = df.drop('liver_cancer', axis=1)
minmax = MinMaxScaler()
norm_df= minmax.fit_transform(df)
```

```
scaled_df = pd.DataFrame(norm_df,columns=df.columns)
```

```
scaled_df.head()
```

	age	gender	bmi	alcohol_consumption	smoking_status	hepatitis_b	hepatitis_c	liver_function_sco
0	0.703704	0.0	0.280899	1.0	0.5	0.0	0.0	0.4210
1	0.944444	0.0	0.331461	0.5	1.0	0.0	0.0	0.3275
2	0.518519	0.0	0.488764	0.0	1.0	0.0	0.0	0.6397
3	0.259259	1.0	0.221910	0.0	0.5	0.0	0.0	0.4065
4	0.777778	1.0	0.362360	0.5	0.5	0.0	0.0	0.3085

```
# Save the scaler as a .pkl file
joblib.dump(minmax, "minmax_scaler.pkl")
# Download the file in Colab
files.download("minmax_scaler.pkl")
```

```
print(y.value_counts())
```

```
liver_cancer
0    3911
1    1089
Name: count, dtype: int64
```

```
scaled_df['liver_cancer'] = y
X = scaled_df.drop(['liver_cancer'],axis=1)
```

```
smote = SMOTE()
X_resampled,y_resempled = smote.fit_resample(X,y)
```

```
df_resampled = pd.concat([X_resampled,y_resempled],axis =1)
```

```
print(df_resampled['liver_cancer'].value_counts())
```

```
liver_cancer
0    3911
1    3911
Name: count, dtype: int64
```

```
p = []
r = []
```

```
f = []
h = []
for i in np.arange(0,100):
    X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.3,random_state=i)
    rf = RandomForestClassifier(random_state=i)
    rf.fit(X_train, y_train)
    y_pred = rf.predict(X_test)
    acc = accuracy_score(y_test, y_pred)
    precision = precision_score(y_test, y_pred)
    recall = recall_score(y_test, y_pred)
    f1 = f1_score(y_test, y_pred)
    h.append(acc)
    p.append(precision)
    r.append(recall)
    f.append(f1)
print(max(h), np.mean(h))

index = h.index(max(h))
print(index)
print("Precision:", p[index])
print("Recall:", r[index])
print("F1:", f[index])
```

```
0.9453333333333334 0.9258466666666668
32
Precision: 0.9772727272727273
Recall: 0.7724550898203593
F1: 0.862876254180602
```

```
X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.3,random_state=32)
```

```
rf = RandomForestClassifier(random_state=32)
```

```
rf.fit(X_train, y_train)
y_pred = rf.predict(X_test)
```

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

```
Accuracy:  0.9453333333333334
```

```
print(df.columns.tolist())
```

```
['age', 'gender', 'bmi', 'alcohol_consumption', 'smoking_status', 'hepatitis_b', 'hepatitis_c', 'liver_functic
```

```
# Save the model as .pkl
joblib.dump(rf, "random_forest_model.pkl")
# Download in Colab
files.download("random_forest_model.pkl")
```

```
!pip install pyngrok
!pip install Flask flask-ngrok
```

```
Collecting pyngrok
  Downloading pyngrok-7.3.0-py3-none-any.whl.metadata (8.1 kB)
Requirement already satisfied: PyYAML>=5.1 in /usr/local/lib/python3.12/dist-packages (from pyngrok) (6.0.2)
Downloading pyngrok-7.3.0-py3-none-any.whl (25 kB)
Installing collected packages: pyngrok
Successfully installed pyngrok-7.3.0
Requirement already satisfied: Flask in /usr/local/lib/python3.12/dist-packages (3.1.1)
```

```
Collecting flask-ngrok
  Downloading flask_ngrok-0.0.25-py3-none-any.whl.metadata (1.8 kB)
Requirement already satisfied: blinker>=1.9.0 in /usr/local/lib/python3.12/dist-packages (from Flask) (1.9.0)
Requirement already satisfied: click>=8.1.3 in /usr/local/lib/python3.12/dist-packages (from Flask) (8.2.1)
Requirement already satisfied: itsdangerous>=2.2.0 in /usr/local/lib/python3.12/dist-packages (from Flask) (2.
Requirement already satisfied: jinja2>=3.1.2 in /usr/local/lib/python3.12/dist-packages (from Flask) (3.1.6)
Requirement already satisfied: markupsafe>=2.1.1 in /usr/local/lib/python3.12/dist-packages (from Flask) (3.0.
Requirement already satisfied: werkzeug>=3.1.0 in /usr/local/lib/python3.12/dist-packages (from Flask) (3.1.3)
Requirement already satisfied: requests in /usr/local/lib/python3.12/dist-packages (from flask-ngrok) (2.32.4)
Requirement already satisfied: charset_normalizer<4,>=2 in /usr/local/lib/python3.12/dist-packages (from reque
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.12/dist-packages (from requests->flask-r
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.12/dist-packages (from requests->f
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.12/dist-packages (from requests->f
Downloading flask_ngrok-0.0.25-py3-none-any.whl (3.1 kB)
Installing collected packages: flask-ngrok
Successfully installed flask-ngrok-0.0.25
```

```
import os

# Create the 'templates' directory if it doesn't exist
if not os.path.exists('templates'):
    os.makedirs('templates')

# Define HTML template
html_template = '''
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Liver Cancer Prediction</title>
    <style>
        body {
            font-family: Arial, sans-serif;
            margin: 20px;
            padding: 20px;
        }
        .container {
            max-width: 600px;
            margin: 0 auto;
            padding: 20px;
            border: 1px solid #ccc;
            border-radius: 10px;
            box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
        }
        input, button, select {
            display: block;
            margin: 10px 0;
            width: 100%;
            padding: 10px;
            box-sizing: border-box;
            border: 1px solid #ccc;
            border-radius: 5px;
        }
        button {
            background-color: #007bff;
            color: white;
            border: none;
            cursor: pointer;
            border-radius: 5px;
        }
        button:hover {
            background-color: #0056b3;
        }
        #result {
```

```
                margin-top: 20px;
            }
            button.print-btn {
                margin-top: 20px;
                background-color: #28a745;
            }
            button.print-btn:hover {
                background-color: #218838;
            }
        </style>
    </head>
    <body>
        <div class="container">
            <h1>Liver Cancer Prediction</h1>
            <form id="predictForm">
                <label for="age">Age:</label>
                <input type="number" id="age" name="age" required>

                <label for="gender">Gender:</label>
                <select id="gender" name="gender" required>
                    <option value="male">Male</option>
                    <option value="female">Female</option>
                </select>

                <label for="bmi">BMI:</label>
                <input type="number" step="any" id="bmi" name="bmi" required>

                <label for="alcohol_consumption">Alcohol Consumption:</label>
                <select id="alcohol_consumption" name="alcohol_consumption" required>
                    <option value="regular">Regular</option>
                    <option value="occasional">Occasional</option>
                    <option value="never">Never</option>
                </select>

                <label for="smoking_status">Smoking Status:</label>
                <select id="smoking_status" name="smoking_status" required>
                    <option value="never">Never</option>
                    <option value="former">Former</option>
                    <option value="current">Current</option>
                </select>

                <label for="hepatitis_b">Hepatitis B (0=No, 1=Yes):</label>
                <select id="hepatitis_b" name="hepatitis_b" required>
                    <option value="0">0</option>
                    <option value="1">1</option>
                </select>

                <label for="hepatitis_c">Hepatitis C (0=No, 1=Yes):</label>
                <select id="hepatitis_c" name="hepatitis_c" required>
                    <option value="0">0</option>
                    <option value="1">1</option>
                </select>

                <label for="liver_function_score">Liver Function Score:</label>
                <input type="number" step="any" id="liver_function_score" name="liver_function_score" required>

                <label for="alpha_fetoprotein_level">Alpha-Fetoprotein Level:</label>
                <input type="number" step="any" id="alpha_fetoprotein_level" name="alpha_fetoprotein_level" required>

                <label for="cirrhosis_history">Cirrhosis History (0=No, 1=Yes):</label>
                <select id="cirrhosis_history" name="cirrhosis_history" required>
                    <option value="0">0</option>
                    <option value="1">1</option>
                </select>
```

```
            <label for="family_history_cancer">Family History of Cancer (0=No, 1=Yes):</label>
            <select id="family_history_cancer" name="family_history_cancer" required>
                <option value="0">0</option>
                <option value="1">1</option>
            </select>

            <label for="physical_activity_level">Physical Activity Level:</label>
            <select id="physical_activity_level" name="physical_activity_level" required>
                <option value="low">Low</option>
                <option value="moderate">Moderate</option>
                <option value="high">High</option>
            </select>

            <label for="diabetes">Diabetes (0=No, 1=Yes):</label>
            <select id="diabetes" name="diabetes" required>
                <option value="0">0</option>
                <option value="1">1</option>
            </select>

            <button type="button" onclick="predict()">Predict</button>
        </form>
        <div id="result">
            <h2>Prediction Result:</h2>
            <p id="prediction"></p>
        </div>
        <button class="print-btn" onclick="window.print()">Print This Page</button>
    </div>

    <script>
        async function predict() {
            const form = document.getElementById('predictForm');
            const formData = new FormData(form);

            const response = await fetch('/predict', {
                method: 'POST',
                body: formData
            });
            const result = await response.json();
            document.getElementById('prediction').innerText =
                result.error ? result.error : 'Prediction: ' + result.prediction;
        }
    </script>
</body>
</html>
'''

# Save the HTML template to a file
with open('templates/index.html', 'w') as f:
    f.write(html_template)



import joblib
import pandas as pd
from flask import Flask, request, jsonify, render_template
from pyngrok import ngrok

# Load saved artifacts
scaler = joblib.load('minmax_scaler.pkl')
rf_model = joblib.load('random_forest_model.pkl')

# Initialize Flask app
app = Flask(__name__)

# Set up ngrok tunnel
ngrok.set_auth_token("31jBbcpMlTVUHbbIpo7tvFEnzAB_4sYYrfhqtMwYWcAxhzm")
```

```
port = 5000
public_url = ngrok.connect(port)
print(f"Ngrok Tunnel URL: {public_url}")

# Hardcoded mappings for categorical columns
categorical_mapping = {
    'gender': {'male': 0, 'female': 1},
    'alcohol_consumption': {'never': 0, 'occasional': 1, 'regular': 2},
    'smoking_status': {'never': 0, 'former': 1, 'current': 2},
    'physical_activity_level': {'low': 0, 'moderate': 1, 'high': 2}
}

# Main route for HTML page
@app.route('/')
def index():
    return render_template('index.html')

# Prediction route
@app.route('/predict', methods=['POST'])
def predict():
    try:
        # Collect input features from form
        input_data = {
            'age': float(request.form['age']),
            'gender': request.form['gender'],
            'bmi': float(request.form['bmi']),
            'alcohol_consumption': request.form['alcohol_consumption'],
            'smoking_status': request.form['smoking_status'],
            'hepatitis_b': int(request.form['hepatitis_b']),
            'hepatitis_c': int(request.form['hepatitis_c']),
            'liver_function_score': float(request.form['liver_function_score']),
            'alpha_fetoprotein_level': float(request.form['alpha_fetoprotein_level']),
            'cirrhosis_history': int(request.form['cirrhosis_history']),
            'family_history_cancer': int(request.form['family_history_cancer']),
            'physical_activity_level': request.form['physical_activity_level'],
            'diabetes': int(request.form['diabetes'])
        }

        # Convert to DataFrame
        df = pd.DataFrame([input_data])

        # Apply hardcoded mappings for categorical features
        for col, mapping in categorical_mapping.items():
            df[col] = df[col].map(mapping)

        # Scale features
        df_scaled = scaler.transform(df)

        # Predict using RandomForest
        prediction = rf_model.predict(df_scaled)[0]

        # Optional: convert 0/1 to readable label
        result_label = 'No Liver Cancer' if prediction == 0 else 'Liver Cancer Risk'

        return jsonify({'prediction': result_label})

    except Exception as e:
        return jsonify({'error': str(e)})

# Run Flask app
if __name__ == '__main__':
    app.run(port=port)
```

```
Ngrok Tunnel URL: NgrokTunnel: "https://b07839a41e79.ngrok-free.app" -> "http://localhost:5000"
 * Serving Flask app '__main__'
```

```
 * Debug mode: off
INFO:werkzeug:WARNING: This is a development server. Do not use it in a production deployment. Use a productic
 * Running on http://127.0.0.1:5000
INFO:werkzeug:Press CTRL+C to quit
INFO:werkzeug:127.0.0.1 - - [25/Aug/2025 09:16:28] "GET / HTTP/1.1" 200 -
INFO:werkzeug:127.0.0.1 - - [25/Aug/2025 09:16:29] "GET /favicon.ico HTTP/1.1" 404 -
/usr/local/lib/python3.12/dist-packages/sklearn/utils/validation.py:2739: UserWarning: X does not have valid f
  warnings.warn(
INFO:werkzeug:127.0.0.1 - - [25/Aug/2025 09:18:34] "POST /predict HTTP/1.1" 200 -
/usr/local/lib/python3.12/dist-packages/sklearn/utils/validation.py:2739: UserWarning: X does not have valid f
  warnings.warn(
INFO:werkzeug:127.0.0.1 - - [25/Aug/2025 09:18:39] "POST /predict HTTP/1.1" 200 -
INFO:werkzeug:127.0.0.1 - - [25/Aug/2025 10:20:55] "GET / HTTP/1.1" 200 -
/usr/local/lib/python3.12/dist-packages/sklearn/utils/validation.py:2739: UserWarning: X does not have valid f
  warnings.warn(
INFO:werkzeug:127.0.0.1 - - [25/Aug/2025 10:23:58] "POST /predict HTTP/1.1" 200 -
/usr/local/lib/python3.12/dist-packages/sklearn/utils/validation.py:2739: UserWarning: X does not have valid f
  warnings.warn(
INFO:werkzeug:127.0.0.1 - - [25/Aug/2025 10:25:17] "POST /predict HTTP/1.1" 200 -
```

Start coding or generate with AI.