```
pip install flask
```

```
Requirement already satisfied: flask in /usr/local/lib/python3.7/dist-packages (1.1.2)
Requirement already satisfied: click>=5.1 in /usr/local/lib/python3.7/dist-packages (fro
Requirement already satisfied: itsdangerous>=0.24 in /usr/local/lib/python3.7/dist-packa
Requirement already satisfied: Werkzeug>=0.15 in /usr/local/lib/python3.7/dist-packages
Requirement already satisfied: Jinja2>=2.10.1 in /usr/local/lib/python3.7/dist-packages
Requirement already satisfied: MarkupSafe>=0.23 in /usr/local/lib/python3.7/dist-package
```

```
from flask import Flask

app = Flask(__name__)


@app.route("/")
def hello():
    return "Welcome to the World of Data Science"


if __name__ == '__main__':
    app.run(debug=True, use_reloader=False)
```

```
 * Serving Flask app "__main__" (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: on
 * Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
```

```
# Import dependencies
import pandas as pd
import numpy as np


# Load the dataset in a dataframe object and include only four features as mentioned
url = "http://s3.amazonaws.com/assets.datacamp.com/course/Kaggle/train.csv"
df = pd.read_csv(url)
include = ['Age', 'Sex', 'Embarked', 'Survived'] # Only four features
df_ = df[include]


# Data Preprocessing
categoricals = []
for col, col_type in df_.dtypes.iteritems():
     if col_type == 'O':
          categoricals.append(col)
     else:
          df_[col].fillna(0, inplace=True)
```

```
/usr/local/lib/python3.7/dist-packages/pandas/core/series.py:4536: SettingWithCopyWarnir
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_
  downcast=downcast,
```

```
df_ohe = pd.get_dummies(df_, columns=categoricals, dummy_na=True)
```

```
# Logistic Regression classifier
from sklearn.linear_model import LogisticRegression
dependent_variable = 'Survived'
x = df_ohe[df_ohe.columns.difference([dependent_variable])]
y = df_ohe[dependent_variable]
lr = LogisticRegression()
lr.fit(x, y)
```

```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, l1_ratio=None, max_iter=100,
                   multi_class='auto', n_jobs=None, penalty='l2',
                   random_state=None, solver='lbfgs', tol=0.0001, verbose=0,
                   warm_start=False)
```

```
# Save your model
from sklearn.externals import joblib
joblib.dump(lr, 'model.pkl')
print("Model dumped!")
```

```
Model dumped!
/usr/local/lib/python3.7/dist-packages/sklearn/externals/joblib/__init__.py:15: FutureWa
  warnings.warn(msg, category=FutureWarning)
```

```
# Load the model that you just saved
lr = joblib.load('model.pkl')

# Saving the data columns from training
model_columns = list(x.columns)
joblib.dump(model_columns, 'model_columns.pkl')
print("Models columns dumped!")
```

```
Models columns dumped!
```

```
# Dependencies
from flask import Flask, request, jsonify
from sklearn.externals import joblib
import traceback
import pandas as pd
import numpy as np
```

```
# Your API definition
app = Flask(__name__)

@app.route('/predict', methods=['POST'])
def predict():
    if lr:
        try:
            json_ = request.json
            print(json_)
            query = pd.get_dummies(pd.DataFrame(json_))
            query = query.reindex(columns=model_columns, fill_value=0)

            prediction = list(lr.predict(query))

            return jsonify({'prediction': str(prediction)})

        except:

            return jsonify({'trace': traceback.format_exc()})
    else:
        print ('Train the model first')
        return ('No model here to use')

if __name__ == '__main__':
    try:
        port = int(sys.argv[1]) # This is for a command-line input
    except:
        port = 12345 # If you don't provide any port the port will be set to 12345

    lr = joblib.load("model.pkl") # Load "model.pkl"
    print ('Model loaded')
    model_columns = joblib.load("model_columns.pkl") # Load "model_columns.pkl"
    print ('Model columns loaded')

    app.run(port=port, debug=True,use_reloader=False)
```

```
Model loaded
Model columns loaded
 * Serving Flask app "__main__" (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: on
 * Running on http://127.0.0.1:12345/ (Press CTRL+C to quit)
```