Steps:-

1. Utilized Google Colab for the setup and training of the model.
2. Created an Anaconda environment dedicated to the project.
3. Developed a website using the Streamlit tool within the Spyder IDE. This website employs the trained model that was initially downloaded from Google Colab.
4. Executed the command "streamlit run "D:\Programming\Project_Vitiligo_SVM_ML\vitiligo_prediction.py"" within the Anaconda terminal of the respective environment to launch the Spyder Python file.

Google colaboratory:-

- free Jupyter notebook environment
- runs entirely in the cloud
- supports many popular machine learning libraries
- code in Python
- Import/Publish notebooks from GitHub
- Import external datasets e.g. from Kaggle
- Colab provides virtual CPU, RAM and also GPU(Graphics Processing Units) hardware for parallel processing for ML

Jupiter notebook:-

- popular tool for data science, machine learning
- The Jupyter kernel: This is the engine that executes the code in your notebook.
- The Jupyter notebook interface
- notebook is divided into cells
- To execute a cell, you can press Shift+Enter
- combine code, text, and visualizations in a single document

Google Colab is a Jupyter notebook. It is a hosted Jupyter notebook environment that runs on Google's servers. This means that you can use Google Colab without having to install any software on your own computer.

Dataset:-

dataset is a collection of data that is used to train the model. A dataset acts as an example to teach the machine learning algorithm how to make predictions.

here are 2 key purposes of an AI/ML dataset:

1. To train the model
2. To measure the accuracy of the model once it is trained

What did we do in the project on google colab?

building and evaluating a Support Vector Machine (SVM) model

1. Importing Dependencies:
   Importing the necessary libraries, including numpy, pandas for data handling, StandardScaler for data preprocessing, train_test_split for splitting the dataset, svm for the Support Vector Machine classifier, and accuracy_score for model evaluation.

   numPy-It provides support for large, multi-dimensional arrays and matrices, along with a wide range of mathematical functions to operate on these arrays.

   Pandas- Pandas is an open-source Python library that provides data manipulation and analysis tools. Pandas is built on top of the NumPy library and offers data structures and functions that simplify working with tabular, labeled, and time-series data.

2. Data Collection and Analysis:
   Loading a dataset from a CSV file using pd.read_csv. The dataset contains information about various features related to vitiligo patches. The code then displays information about the dataset, such as its shape, statistical measures, and the count of different outcomes (vitiligo patch or not).

3. Data Preprocessing:
   The code separates the dataset into features (X) and labels (Y). The features are standardized using StandardScaler to make the data comparable across different scales.

4. Train-Test Split:
   The dataset is split into training and testing sets using train_test_split in an 80:20 ratio. Stratification is used to maintain the distribution of outcomes in both sets.

5. Training the Model:
   An SVM classifier with a linear kernel is initialized and trained on the training data using the fit method.

6. Model Evaluation:
   The accuracy of the trained model is evaluated using the accuracy_score (function) metric on both the training and testing data. The accuracy shows that the model is not over trained(the accuracy for training data and test data is nearly the same as 70%)

7. Making a Predictive System:
   A hypothetical input patch is standardized and fed into the trained model to predict whether it's a vitiligo patch or not. The result of the prediction is displayed.

8. Saving the Trained Model:
   The trained model is saved to a file using the pickle library. The saved model is then loaded, and the same predictive process is repeated using the loaded model.

Training the model (detailed explanation):-

```
classifier = svm.SVC(kernel='linear')
```

```
# Training the support vector machine Classifier
classifier.fit(X_train, Y_train)
```

Classifier Initialization:
In this step, a Support Vector Machine (SVM) classifier is created. The svm.SVC class is used to initialize the classifier. The parameter kernel='linear' specifies that a linear kernel will be used for the SVM. The kernel determines how the SVM separates the data points in a higher-dimensional space.

Linear Kernel- It's a mathematical function that defines the similarity between data points

Training Data:
The X_train and Y_train variables contain the training features and labels, respectively. The X_train variable contains the standardized features of the training data, and Y_train contains the corresponding labels indicating whether a patch is vitiligo (1) or not (0).

Model Training:
The fit method is used to train the classifier using the training data. The classifier learns how to separate the data points into different classes based on the provided features and labels. It determines the optimal hyperplane that best separates the two classes in the feature space.

Hyperplane- A hyperplane is a flat decision boundary that is used to separate data points belonging to different classes in a feature space.

Support Vector Machine (SVM):

SVM is a supervised machine learning algorithm primarily used for classification and regression tasks. Its primary objective is to find a hyperplane that best separates data points of different classes in a feature space.
(Everything is in two-dimensional space)

Classification tasks-Classification tasks are a type of supervised machine learning problem where the goal is to assign predefined labels or categories to input data points.

Regression tasks- Regression tasks are a type of supervised machine learning problem where the goal is to predict a continuous numerical value or quantity based on input features.

supervised machine learning problem- A supervised machine learning problem is a type of machine learning task where the algorithm learns from labeled training data to make predictions or decisions on new, unseen data. In supervised learning, the algorithm is "supervised" during training by having access to both input features and their corresponding target labels or outcomes. The goal is to learn a mapping from input features to output labels in order to make accurate predictions on new data points.

Basic Idea:-
The basic idea behind SVM is to find the hyperplane that maximizes the margin between data points of different classes. The margin is the distance between the hyperplane and the closest data points from each class. SVM aims to find the hyperplane that has the maximum margin while still correctly classifying data points.

Anaconda:-

- open-source platform
- field of data science and scientific computing
- setting up and managing environments
- comes with python and pre installed packages and tools
- Anaconda provides the "conda" package manager, which allows users to easily install, update, and manage packages for various programming languages

An environment refers to an isolated space where a specific set of software components, libraries, and dependencies are installed and configured to work together. Environments are created to avoid conflicts between different projects

Spyder:-

Spyder, which stands for "Scientific PYthon Development EnviRonment," is an open-source integrated development environment (IDE) primarily designed for scientific computing, data analysis, and research using the Python programming language.

In spyder IDE we have written code for Streamlit web application that uses a machine learning model to predict whether an observed skin patch is indicative of vitiligo or not.

1. Importing Libraries:
   The pickle library is imported to load the pre-trained machine learning model.
   The streamlit library is imported to create the interactive web application.

2. Loading the Model:
   The pre-trained SVM machine learning model for predicting vitiligo is loaded using the pickle.load() function.

3. Creating the Web Application:
   The st.title() function displays the title of the web application.

4. Gathering Input Data:
   Text input fields are created for the user to input various medical parameters related to the skin patch observation, such as Glucose levels, Blood Pressure, Skin Thickness, BMI, Hemoglobin Levels, Stress Levels, and Age. These inputs are organized into three columns using st.columns() for better layout.

5. Prediction Logic:
   The code block for prediction is set up, and an empty string named vitiligo_diagnosis is initialized to store the diagnosis result.

6. Button for Prediction:
   A button is created using st.button() with the label "Vitiligo Test Result." When the button is clicked, the prediction process is triggered.

7. Prediction Process:

If any input field is left empty, a warning message is displayed using st.warning() to prompt the user to fill in all the fields.

If all input fields are filled, the machine learning model predicts whether the observed patch is vitiligo or not by passing the input values to the model using vitiligo_model.predict().

Depending on the prediction result, the diagnosis is assigned to vitiligo_diagnosis as either "This person's Observed Patch is Vitiligo" or "This person's Observed Patch is Not Vitiligo."

Streamlit:-

- open-source Python library
- creating interactive web applications for data science and machine learning projects
- data scripts into shareable web apps quickly

Streamlit itself doesn't directly provide a URL for your website. Instead, it's a tool that helps you create interactive web applications using Python scripts. To make your Streamlit app accessible online with a URL, you need to deploy it on a web server or a cloud platform.