Disease Prediction on Breast Cancer

Master Of Science

In

Computer And Information Science

420 University Blvd, Indianapolis, IN 46202

Group No: 5

Student Id: 2001018024
Name : Sowmya Nuchu
Email: snuchu@iu.edu

Student Id: 2001131345
Name : Jinal Patel
Email : pateljin@iu.edu

TABLE OF CONTENTS

ABSTRACT:

The recent high population expansion in medical research has made early disease identification an urgent problem. With this increase in population comes an exponential rise in the danger of dying from breast cancer. A system for automatically detecting diseases lowers the chance of death and enables medical personnel to respond in a timely, accurate, and dependable manner. In this task, we will examine the data using four supervised machine learning techniques (support vector machine (SVM), K-nearest neighbors, random forests, and decision tree classifier) to determine from this assessment whether the cancer is benign or malignant. A well-known platform called Kaggle enables access to the Real breast cancer sample dataset collection, which outlines a group of breast cancer patients who undergone surgical removal of tumors. Computing precision, recall and accuracy will be used to distinguish the algorithms' efficiency.

METHODOLOGY FOLLOWED:

The steps followed to do this project are:

- Collection of dataset.
- Understanding features of dataset.
- Pre-processing the data.
- Split data into training dataset and testing dataset.
- Apply ML algorithms to dataset to predict the breast cancer.
- Improving results

FUNCTIONAL REQUIREMENTS

The software’s are defined by the functional requirements. Basically, intakes, actions, and results make up a program. Computation, specific details, data processing, and data modification are instances of objects that belong to the category of acceptance criteria.

The program here carries out the following tasks:

- Recognize each characteristic and the information provided in the dataset.
- With the source information given, map the dataset's data. With the dataset and the data sources, investigate trends, if any.
- If breast cancer is discovered, describe the nature, such as whether it is benign or malignant.
- Indicate the prediction's accuracy in percentage.

DESIGN

As per our framework, the objective of our task is to create a solution that will enable us to:

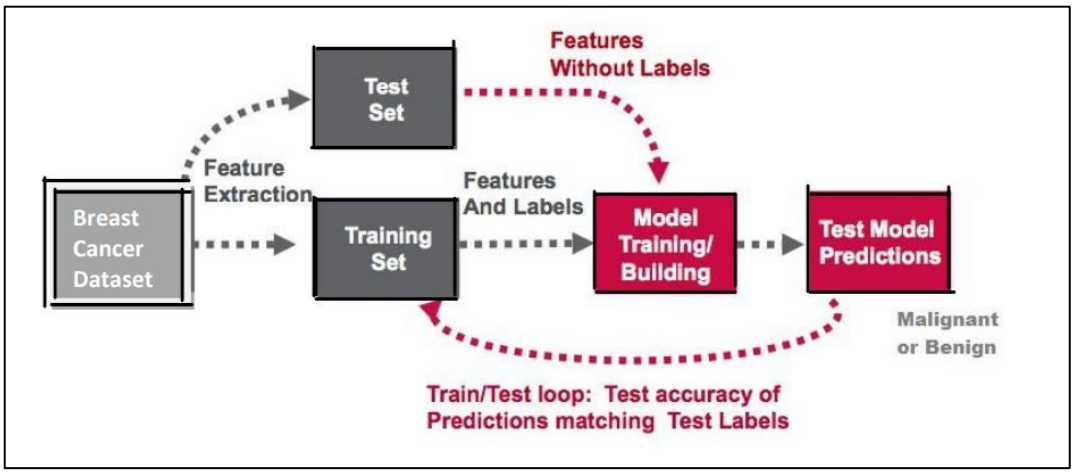
I. Accuracy

Only appropriate results will make this set of parameters useful, o Only when all of the results are accurate and believable It is crucial that there are no inaccuracies because this data is essential for investigation and treatment.

II. Efficiency

Given that there is no need for manual data entry or medical work, the concept ought to be effective. After all the Machine learning algorithms have been trained to the data, estimation durations are reduced.

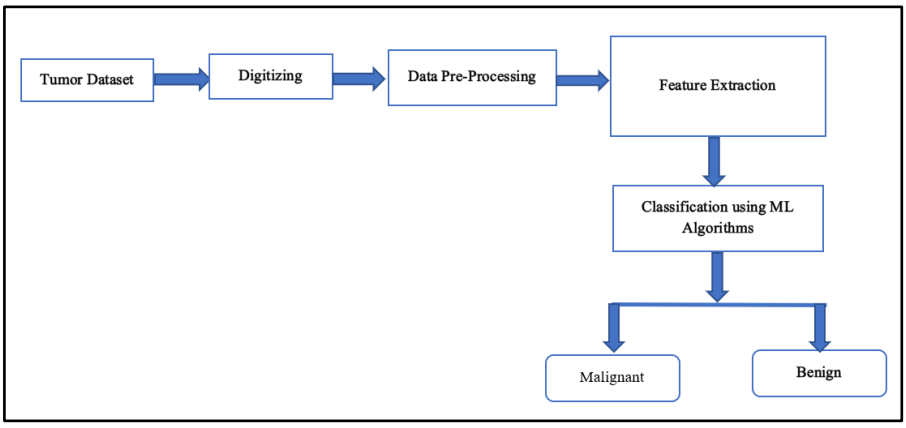
SYSTEM ARCHITECTURE



- The layout of this project is basically the dataset and its characteristics because it lacks any UI. The system is being simplified and made as user-friendly as it can be easy in understanding the dataset.
- First, the dataset is divided into a training set and a testing set. The machine learning algorithms are first introduced to the training set so that the machine can learn which data produces which kinds of results.
- The testing information is used to analyze whether the system can accurately estimate the class of the data once it has been trained. It examines the model's 100 % accuracy.

EXPECTED OUTCOME:

This model's output is to check accurate detection and prediction of whether a participant has cancer or not by using machine learning algorithms. If the response is positive, the model ought to be able to determine whether the person has malignant or benign cancer.



IMPLEMENTAION AND OUTPUT

Analyze the features of the Dataset

STEP 1: Organizing the data is the initial step in the machine learning process. Installing all the modules that will facilitate data management and visualization is part of this process. The below are the used packages:

Loading the libraries and the dataset

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

# Modeling

from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.manifold import TSNE
from sklearn.preprocessing import StandardScaler

import warnings
warnings.filterwarnings('ignore')


# Pre-Modeling Tasks

from sklearn.preprocessing import LabelEncoder
from sklearn.decomposition import PCA
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler

# Evaluation and comparision of all the models


from sklearn.metrics import accuracy_score
from sklearn.metrics import classification_report
from sklearn.metrics import precision_score, recall_score, confusion_matrix, precision_recall_fscore_support
from sklearn.metrics import roc_auc_score,auc,f1_score
from sklearn.metrics import precision_recall_curve,roc_curve
```

STEP II: The dataset must be loaded once all essential modules have been imported. In order to load the dataset we use pandas.

In [2]:
```
tumor = pd.read_csv("data.csv")
df=pd.DataFrame(tumor);
```

STEP III: We look at the dataset and analyze the features and types of attributes present in the dataset.

Ten real-valued features are computed for each cell nucleus:

a) radius (mean of distances from center to points on the perimeter)
b) texture (standard deviation of gray-scale values)
c) perimeter
d) area
e) smoothness (local variation in radius lengths)
f) compactness (perimeter^2 / area - 1.0)
g) concavity (severity of concave portions of the contour)
h) concave points (number of concave portions of the contour)
i) symmetry
j) fractal dimension ("coastline approximation" - 1)

Attribute Information:

- ID number
- Diagnosis (M = malignant, B = benign)

We have total 357 values for Benign and 212 for Malignant

```
In [6]: df['diagnosis'].value_counts()

Out[6]: B    357
        M    212
        Name: diagnosis, dtype: int64
```

```
In [6]: tumor.shape

Out[6]: (569, 33)
```

```
In [5]: tumor.columns

Out[5]: Index(['id', 'diagnosis', 'radius_mean', 'texture_mean', 'perimeter_mean',
               'area_mean', 'smoothness_mean', 'compactness_mean', 'concavity_mean',
               'concave points_mean', 'symmetry_mean', 'fractal_dimension_mean',
               'radius_se', 'texture_se', 'perimeter_se', 'area_se', 'smoothness_se',
               'compactness_se', 'concavity_se', 'concave points_se', 'symmetry_se',
               'fractal_dimension_se', 'radius_worst', 'texture_worst',
               'perimeter_worst', 'area_worst', 'smoothness_worst',
               'compactness_worst', 'concavity_worst', 'concave points_worst',
               'symmetry_worst', 'fractal_dimension_worst', 'Unnamed: 32'],
              dtype='object')
```

Data Preprocessing

The first step in any machine learning process is data preprocessing, where information is prepared for metrics operations to obtain the most accurate insights. Preprocessing data fills up approximately 90% of the time spent on tasks including data analytics, machine learning and data visualization.

In general, machine learning algorithms cannot work with missing values, so before launching one, the dataset must be cleaned up by removing features that have no bearing on the model. In this case we are removing the id and unnamed field as it will not be used in categorization process.

```
In [13]: # Deleting the id and Unnamed column

         df= df.drop(['Unnamed: 32','id'],axis=1)
```

Checks for missing values

```
In [12]: df.isnull().sum()
```

Exploratory Data Analysis & Data Visualization

After data gathering and preparation, EDA is a fundamental phase where the data is simply presented, projected, and performed without any assumptions to aid in determining the accuracy of the information and creating models. A few statistical approaches supplement most graphical EDA tools.

Basic Statistical Details

```
In [11]: # describing the dataset

         df.describe().T
```

Out[11]:

	count	mean	std	min	25%	50%	75%	max
id	569.0	3.037183e+07	1.250206e+08	8670.000000	869218.000000	906024.000000	8.813129e+06	9.113205e+08
radius_mean	569.0	1.412729e+01	3.524049e+00	6.981000	11.700000	13.370000	1.578000e+01	2.811000e+01
texture_mean	569.0	1.928965e+01	4.301036e+00	9.710000	16.170000	18.840000	2.180000e+01	3.928000e+01
perimeter_mean	569.0	9.196903e+01	2.429898e+01	43.790000	75.170000	86.240000	1.041000e+02	1.885000e+02
area_mean	569.0	6.548891e+02	3.519141e+02	143.500000	420.300000	551.100000	7.827000e+02	2.501000e+03
smoothness_mean	569.0	9.636028e-02	1.406413e-02	0.052630	0.086370	0.095870	1.053000e-01	1.634000e-01
compactness_mean	569.0	1.043410e-01	5.281276e-02	0.019380	0.064920	0.092630	1.304000e-01	3.454000e-01
concavity_mean	569.0	8.879932e-02	7.971981e-02	0.000000	0.029560	0.061540	1.307000e-01	4.268000e-01
concave points_mean	569.0	4.891915e-02	3.880284e-02	0.000000	0.020310	0.033500	7.400000e-02	2.012000e-01
symmetry_mean	569.0	1.811619e-01	2.741428e-02	0.106000	0.161900	0.179200	1.957000e-01	3.040000e-01
fractal_dimension_mean	569.0	6.279761e-02	7.060363e-03	0.049960	0.057700	0.061540	6.612000e-02	9.744000e-02

The schematic diagram of information and records is known as data visualization. Data visualization tools provide a handy approach to observe and comprehend trends, outliers, and patterns in data by utilizing graphic components like charts, graphs, and maps.

- To determine how to move forward with the machine learning algorithms, we must create data visualizations.
- The first visualization method that is used is heat maps. A two-dimensional display of data is called a heat map. Users can understand complex data sets with the aid of heatmaps by using increasingly complex heat maps.

- Pair Plot: We can also see how the malignant or benign tumors cells can have (or not) different values for the features plotting the distribution of each type of diagnosis for each of the mean features.

- Count Plot : A seaborn count plot can be used to show how frequently both categories of the target variable occur. Category B, or benign, has a frequency of 357, whereas category M, or malignant, has a frequency of 212.

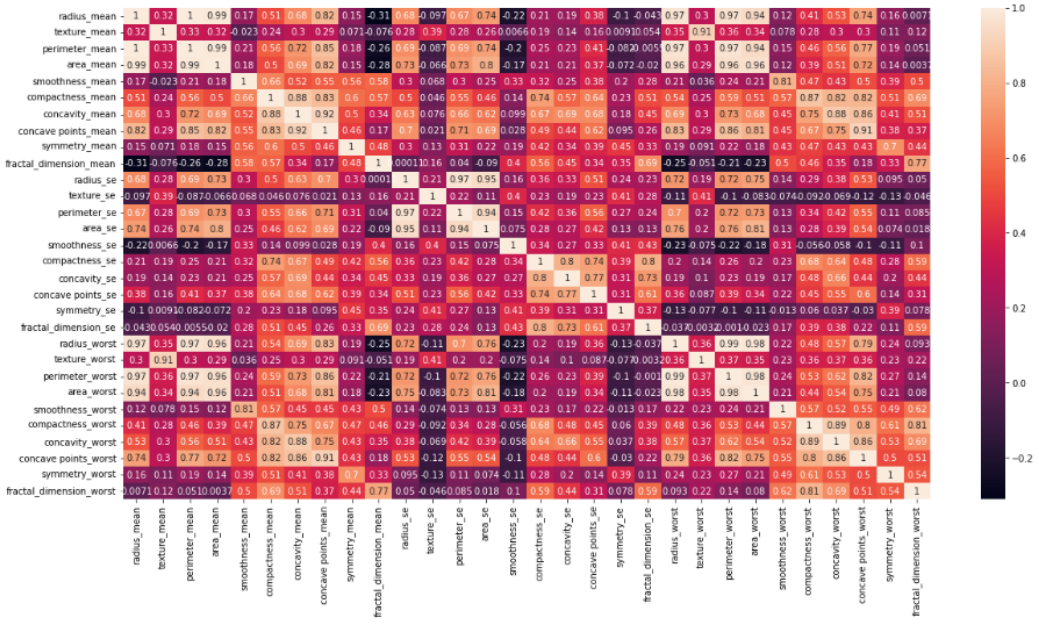
Checking for the correlation

In [11]:
```
features= list(tumor.columns[1:11])
print(features)
```

```
['radius_mean', 'texture_mean', 'perimeter_mean', 'area_mean', 'smoothness_mean', 'compactness_mean', 'concavity_mean', 'co
ncave points_mean', 'symmetry_mean', 'fractal_dimension_mean']
```

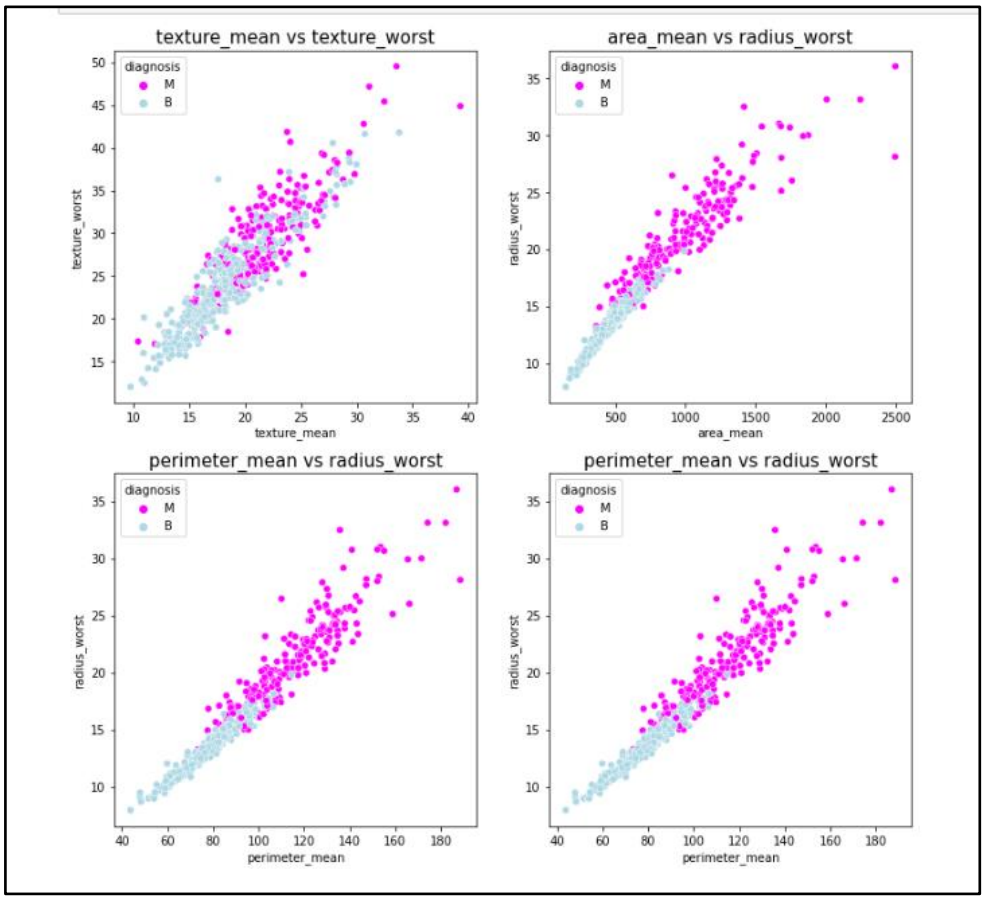
In [12]:
```
plt.figure(figsize=(20,10))
sns.heatmap(tumor.corr(),annot=True)
plt.ioff()
```

Out[12]: `<matplotlib.figure.Figure at 0x1c86dcb5640>`



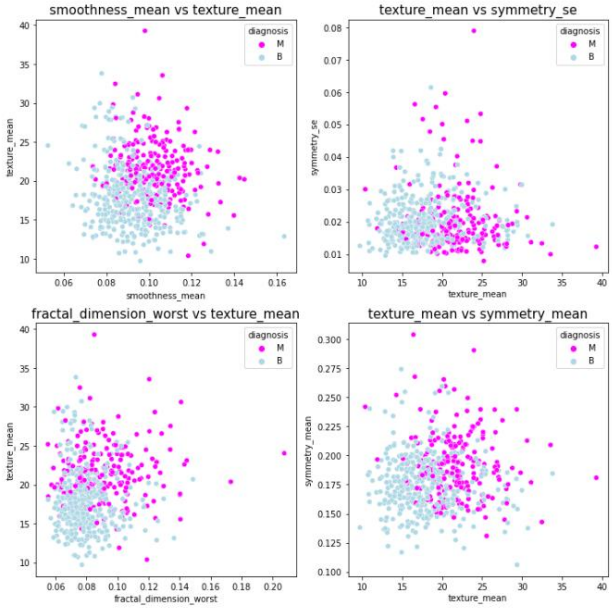
Highly Correlated Pair

In [38]:
```
palette ={'B' : 'lightblue', 'M' : 'magenta'}


fig = plt.figure(figsize=(12,12))
def plot_scatter(a,b,k):
    plt.subplot(k)
    sns.scatterplot(x = df[a], y = df[b], hue = "diagnosis",
                    data = df, palette = palette)
    plt.title(a + ' vs ' + b,fontsize=15)

plot_scatter('texture_mean','texture_worst',221)
plot_scatter('area_mean','radius_worst',222)
plot_scatter('perimeter_mean','radius_worst',223)
plot_scatter('perimeter_mean','radius_worst',224)
```



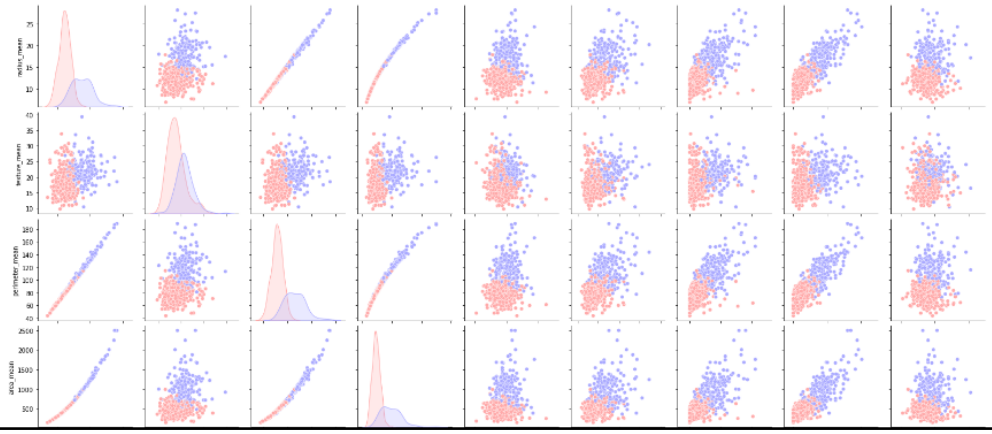
Inverse Correlated Pairs

In [40]:
```
fig = plt.figure(figsize=(12,12))
plot_scatter('smoothness_mean','texture_mean',221)
plot_scatter('texture_mean','symmetry_se',222)
plot_scatter('fractal_dimension_worst','texture_mean',223)
plot_scatter('texture_mean','symmetry_mean',224)
```



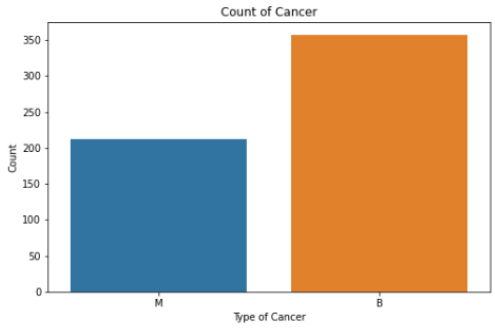
PairPlot

In [13]:
```
from pylab import rcParams

rcParams['figure.figsize'] = 8,5

cols = ['radius_mean', 'texture_mean', 'perimeter_mean',
       'area_mean', 'smoothness_mean', 'compactness_mean', 'concavity_mean',
       'concave points_mean', 'symmetry_mean','diagnosis']

sns_plot = sns.pairplot(data=tumor[cols],hue='diagnosis', palette='bwr')
```



Count Plot

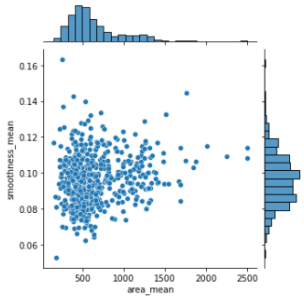
In [14]:
```
# Target variable
plt.title('Count of Cancer')
sns.countplot(tumor['diagnosis'])
plt.xlabel('Type of Cancer')
plt.ylabel('Count')
plt.show()
```



JointPlot

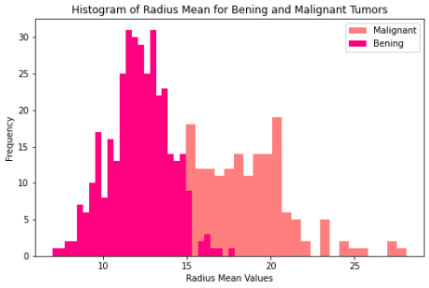
In [16]:
```
sns.jointplot(data= tumor, x='area_mean', y='smoothness_mean', size=5)
```

Out[16]: <seaborn.axisgrid.JointGrid at 0x1c8739a8d90>



Histogram

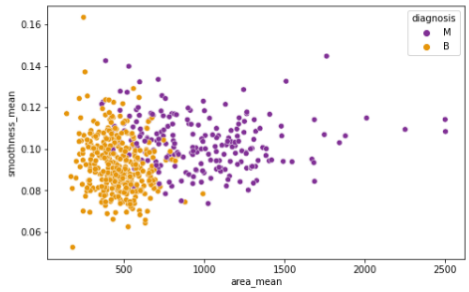
In [15]:
```
m = plt.hist(df[tumor["diagnosis"] == "M"].radius_mean,bins=30,fc = (1,0,0,0.5),label = "Malignant")
b = plt.hist(df[tumor["diagnosis"] == "B"].radius_mean,bins=30, fc = (1,0,0.5), label= "Bening")

plt.legend()
plt.xlabel ("Radius Mean Values")
plt.ylabel ("Frequency")
plt.title("Histogram of Radius Mean for Bening and Malignant Tumors")
plt.show()
```



Scatter Plot

In [38]:
```
sns.scatterplot(x= 'area_mean', y= 'smoothness_mean', hue= 'diagnosis', data=df, palette='CMRmap')
```

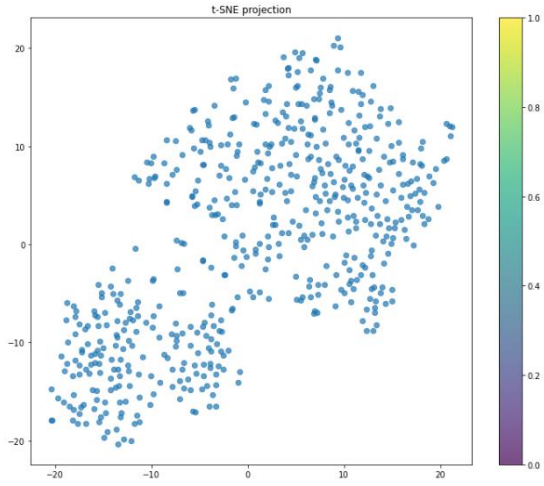
Out[38]: <AxesSubplot:xlabel='area_mean', ylabel='smoothness_mean'>



T-SNE Plot

In [22]:
```
tsne = TSNE(n_components=2, verbose=1, perplexity=40, n_iter=2000,random_state = 35)
df_tsne_scaled = tsne.fit_transform(X_scaled)

plt.figure(figsize=(12,10))
plt.scatter(df_tsne_scaled[:, 0], df_tsne_scaled[:, 1], alpha=0.7, s=40)
plt.colorbar()
plt.title('t-SNE projection');
```

```
[t-SNE] Computing 121 nearest neighbors...
[t-SNE] Indexed 569 samples in 0.000s...
[t-SNE] Computed neighbors for 569 samples in 0.012s...
[t-SNE] Computed conditional probabilities for sample 569 / 569
[t-SNE] Mean sigma: 1.522404
[t-SNE] KL divergence after 250 iterations with early exaggeration: 65.971390
[t-SNE] KL divergence after 1600 iterations: 0.862887
```



Pre-Modelling Tasks

Machine learning algorithms can only interpret numerical numbers, based on what we know. Encoding categorical attributes into numerical values is important

Encoding categorical data

```
In [17]:
# Label Encoder

LEncoder = LabelEncoder()

tumor['diagnosis'] = LEncoder.fit_transform(tumor['diagnosis'])
```

```
Seperating dependent and independent variables

In [19]:
X = tumor.drop('diagnosis',axis=1).values
y = tumor['diagnosis'].values
```

Splitting the dataset

In Machine learning we must split the dataset into training and testing data:

- The training set, also known as the learning set, contains most of the data that we'll need to build our model.
- The Testing set: It is used to assess the model's performance following hyperparameter adjustment. It is also beneficial for evaluating the results of other models.
- SKit_Learn provides a function of splitting the dataset into multiples subsets.
- The simplest method, train test split(), supports lists, numpy arrays, sci-sparse matrices, or pandas dataframes. It is the same as the function split train test().We will also identify some parameters, like the random_state that allows you to set the random generator seed.
- The ideal split is said to be 75:25 for training and testing. You may need to adjust it depending on the size of the dataset and parameter complexity.

Splitting the dataset

```
In [23]:
random_state = 35

x_train, x_test, y_train, y_test = train_test_split(X_scaled,y, test_size=0.25, random_state=random_state)
```

Feature Scaling

A technique for standardizing the variety of independent variables or features in data is feature sizing. Leveling the data is crucial for improving the score.

Data pre-processing step known as "feature scaling" applies independent variables or data features. In essence, it aids in normalizing the data within a specific range.

Feature Scaling

In [24]:
```
fs = StandardScaler()

X_training = fs.fit_transform(x_train)
X_testing= fs.transform(x_test)
```

Applying ML algorithms to dataset to predict breast cancer.

We'll experiment with various machine learning models in this section: Decision tree, Random Forest, Support Vector Machine, and K Nearest Neighbors Model. Standard scaler will normalize the features such that each feature will be having mean as 0 and standard deviation as 1. Then the training data set is given to learning algorithms like logistic regression, random forest, and decision tree to generate models. Now to predict whether the cancer cells are malignant or benign, the test data should be given to the model which was generated by using the training data set

Modeling

In [25]:
```
# Random Forest Classifier

a = RandomForestClassifier()

a.fit(X_training, y_train)

y_pred_rf1 = a.predict(X_testing)
print(y_pred_rf1)


# Decision Tree Classifier

b = DecisionTreeClassifier()

b.fit(X_training, y_train)

y_pred_dt1 = b.predict(X_testing)


# KNeighbors Classifier


c = KNeighborsClassifier(n_neighbors=5)

c.fit(X_training, y_train)


# Support Vector classifier

svc = SVC(probability=True)

svc.fit(X_training,y_train)

y_pred_svc = svc.predict(X_testing)
```

Evaluation and Comparison of All Models

Any data science work must include a thorough assessment of the machine learning algorithms. Numerous metrics enable us to assess the correctness of our models.

- Classification Accuracy
- Confusion matrix
- Precision
- Recall
- Classification_report
- ROC AUC Score

Once the data is processed the training data is provided to the learning algorithms like Random Forest, K-NN, Decision tree and Support vector machine to generate models . The model that was created using the training data set should now be given the to test data in order to determine whether the cancer cells are malignant or benign.

Precision:

In terms of positive observations, precision is the proportion of accurately anticipated observations to all predicted positive observations.

Precision

In [32]:
```
models = []

R = [SVC() , KNeighborsClassifier() , RandomForestClassifier() , DecisionTreeClassifier() ]


X = ["SVM" , "DecisionTreeClassifier" , "KNeighborsClassifier" ,
    "RandomForestClassifier" ]

for i in range(0,len(Z)):
    model = R[i]
    model.fit( X_training , y_train )
    pred = model.predict(X_testing)
    models.append(precision_score(pred , y_test))

d = { "Precision" : models ,  "Algorithm" : X }
data_frame = pd.DataFrame(d)
data_frame
```

Out[32]:

	Precision	Algorithm
0	0.94	SVM
1	0.92	DecisionTreeClassifier
2	0.94	KNeighborsClassifier
3	0.84	RandomForestClassifier

Recall:

The ratio of positive cases that are correctly identified by the classifier to all observations in the real class is known as recall, also known as sensitivity.

Recall

In [30]:
```
models = []

R = [SVC() , KNeighborsClassifier() , RandomForestClassifier() , DecisionTreeClassifier() ]


X = ["SVM" , "DecisionTreeClassifier" , "KNeighborsClassifier" ,"RandomForestClassifier" ]

for i in range(0,len(Z)):
    model = Z[i]
    model.fit( X_training , y_train )
    pred = model.predict(X_testing)
    models.append(recall_score(pred , y_test))
```

In [31]:
```
d = { "Recall" : models ,  "Algorithm" : X }
data_frame = pd.DataFrame(d)
data_frame
```

Out[31]:

	Recall	Algorithm
0	1.000000	SVM
1	0.978723	DecisionTreeClassifier
2	0.958333	KNeighborsClassifier
3	0.914894	RandomForestClassifier

Accuracy : The most understandable performance metric is accuracy score, which is basically the ratio of successfully predicted observations to all observations.

Evaluation and comparison of all the models

In [27]:
```
models = []

Z = [SVC() , KNeighborsClassifier() , RandomForestClassifier() , DecisionTreeClassifier() ]


X = ["SVM" , "DecisionTreeClassifier" , "KNeighborsClassifier" ,"RandomForestClassifier" ]

for i in range(0,len(Z)):
    model = Z[i]
    model.fit( X_training , y_train )
    pred = model.predict(X_testing)
    models.append(accuracy_score(pred , y_test))
```

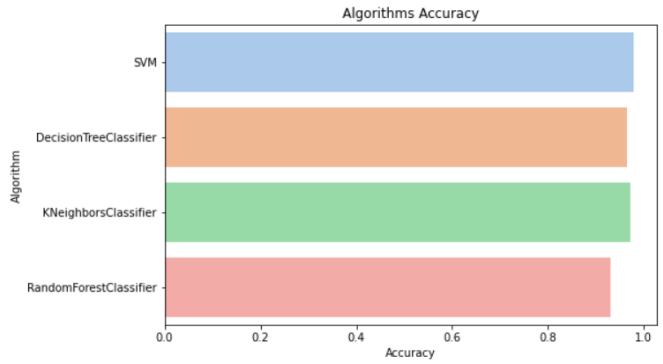
In [28]:
```
d = { "Accuracy" : models ,  "Algorithm" : X }
data_frame = pd.DataFrame(d)
data_frame
```

Out[28]:

	Accuracy	Algorithm
0	0.979021	SVM
1	0.965035	DecisionTreeClassifier
2	0.972028	KNeighborsClassifier
3	0.930070	RandomForestClassifier

Upon applying supervised algorithms we found out that , the SVM classifier is being proved more efficient and accurate for the prediction and classification of the tumor as malignant (cancerous) or benign (non-cancerous).

```
In [29]: sns.barplot(data_frame['Accuracy'],data_frame['Algorithm'],palette= "pastel").set_title('Algorithms Accuracy')
Out[29]: Text(0.5, 1.0, 'Algorithms Accuracy')
```



Evaluation metrics

Algorithms	Precision	Recall	Accuracy
KNN	0.94	0.958333	0.972028
SVM	0.94	1.000000	0.979201
Random Forest	0.84	0.914894	0.930070
Decision Tree	0.92	0.978723	0.965035

Confusion Matrix :

A confusion matrix is a list that can be used to assess how well a machine learning algorithm, often a supervised learning algorithm, is performing. The instances of an actual class are represented by each row, while the instances of a predicted class are represented by each column, in the confusion matrix.

Confusion Matrix

```
In [33]: cm = np.array(confusion_matrix(y_test, y_pred_svc, labels=[1,0]))

         confusion_mat= pd.DataFrame(cm, index = ['cancer', 'healthy'],
                                     columns =['predicted_cancer','predicted_healthy'])

         confusion_mat
```

Out[33]:

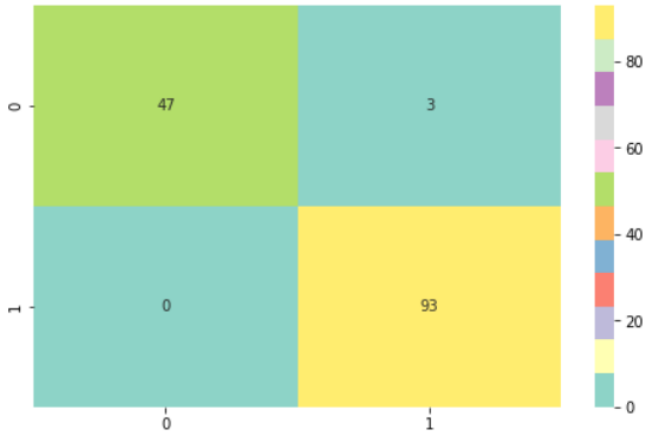
	predicted_cancer	predicted_healthy
cancer	47	3
healthy	0	93

From the below table it is depicted that

- True Positives (TP) : The values that the model predicted would be true (healthy) and are in fact true (Healthy).
- True Negative : The values that the model predicted would be false (Cancer) and is actually False(Cancer)
- False Positive : The values that the model predicted as true(Healthy), but actually is no (Cancer).

- False Negative: The values that the model predicted as false(Cancer), but actually true(Healthy).

```
In [34]: sns.heatmap(cm,annot=True,fmt='g',cmap='Set3')
Out[34]: <AxesSubplot:>
```



Regarding this dataset, if the model predicts something as yes, it shows the absence of cancer cells (Healthy), and when it predicts something as no, it indicates the presence of cancer cells (Cancer).

Classification Report

- True Positive(TP) : 47
- True Negative(TN) : 93
- False Positive(FP): 3
- False Negative(FN): 0

Classification Report

```
In [35]: print(classification_report(y_test, y_pred_svc))
```

	precision	recall	f1-score	support
0	0.97	1.00	0.98	93
1	1.00	0.94	0.97	50
accuracy			0.98	143
macro avg	0.98	0.97	0.98	143
weighted avg	0.98	0.98	0.98	143

True Positive Rate/Recall/Sensitivity: How often the model is predicted as true(Healthy) when it's actually true(Healthy)?

- **True Positive Rate(TPR)** = TP/TP+FP = 47/(47+3) = 0.9591

False Positive Rate: How often the model predicts true(Healthy) when it's actually false (Cancer)?

- **False Positive Rate(FPR)** = FP/FP+TN = 3/96 = 0.031

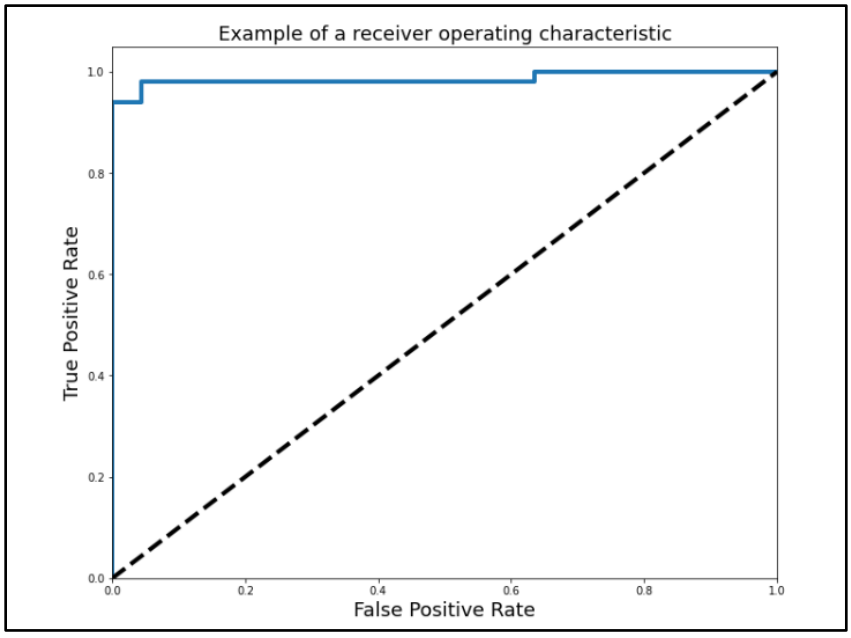
ROC Curve

The ROC curve below illustrates the exchange between specificity and sensitivity (or TPR) (1 – FPR). The svm curve is closer to the top-left corner, which indicates superior performance, as it is shown.

The ROC Curve

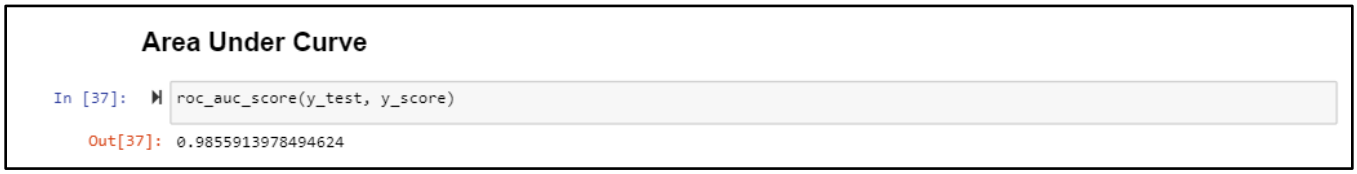
In [36]:
```
y_score = svc.decision_function(X_testing)

FPR, TPR, _ = roc_curve(y_test, y_score)
ROC_AUC = auc(FPR, TPR)
print (ROC_AUC)

plt.figure(figsize =[11,9])
plt.plot(FPR, TPR, label= 'ROC curve(area = %0.2f)'%ROC_AUC, linewidth= 4)
plt.plot([0,1],[0,1], 'k--', linewidth = 4)
plt.xlim([0.0,1.0])
plt.ylim([0.0,1.05])
plt.xlabel('False Positive Rate', fontsize = 18)
plt.ylabel('True Positive Rate', fontsize = 18)
plt.title('Receiver operating characteristic example', fontsize= 18)
plt.show()
```

```
0.9855913978494624
```



Area Under Curve

Comparing classifiers is frequently done using area under the curve. A ideal classifier will have a ROC AUC of 1.



CONCLUSION:

In our Breast Cancer Prediction Project, four classification models namely KNN, Support Vector Machine, Random Forest and Decision Tree were trained and optimized. From the evaluations we get two results of model with exact accuracy of 0.94 that is KNN and SVM. Since it is tumor detection, we need to get the actual sensitivity of the data, so both precision and recall score are important as well. Thus, we calculated precision and recall for all the four models and compared the results with the accuracy rate of them. Based on the outputs, the confusion we had between the KNN and SVM was solved as the Recall rate was perfect 1 for SVM. The result states that there were no false Negative predictions detected for the data. In simple terms, Person having the Malignant(cancerous) is never detected as not having cancerous tumor. So, it concludes that SVM is the best classification model for this data of Breast Cancer Prediction.

ACKNOWLEDGEMENT

I would like to express my sincere gratitude to my mentor **Professor Hasan** and TA **Niloy Talukder** for guiding us throughout this project and providing their valuable feedback and insights. The constant support and motivation throughout the semester has helped me to keep up my spirit and solve challenges.