



Predicting Diabetes using machine learning algorithms

Authors

Haris Chandra Naik Gugulothu
Pravanith reddy Kankanala
Reshmitha Mandava
Sai teja Gajji
Supraja Medicherla
Venkata Sai Pallavi Pallapolu

Table of contents



- Data Description
- Project 1-6
- Data Attributes
- Feature selection
- Data visualizations
- Confusion matrix
- Linear regression
- Elbow graph
- K-Nearest Neighbors
- Naive Bayes classifier
- Support vector machine with 4 kernels
- Random forest classifier
- Conclusion
- References
- Contributions

Data description



- Source:

https://data.world/informatics-edu/diabetesvanderbilt/workspace/file?filename=diabetes_raw_Vanderbilt.csv

- The data consist of 14 variables on 403 subjects from 1046 subjects who were interviewed in a study to understand the prevalence of obesity, diabetes, and other cardiovascular risk factors in central Virginia for African Americans. According to Dr John Hong, Diabetes Mellitus Type II (adult onset diabetes) is associated most strongly with obesity. The waist/hip ratio may be a predictor in diabetes and heart disease. DM II is also associated with hypertension - they may both be part of "Syndrome X". The 403 subjects were the ones who were actually screened for diabetes. Glycosylated hemoglobin > 7.0 is usually taken as a positive diagnosis of diabetes.

Project 1-6



- **Project 1:** Defining the project, finding and collecting the data
- **Project 2:** Feature selection and data visualization
- **Project 3:** Confusion matrix and linear regression
- **Project 4:** K-Nearest Neighbors and Naive Bayes classifier
- **Project 5:** Support Vector Machine with 4 Kernels
- **Project 6:** Random forest classifier



Attributes

- Patient ID
- Total cholesterol
- Random glucose
- HDL cholesterol
- Cholesterol/HDL ratio
- Hemoglobin A1c
- Age

- Gender
- Height
- Weight
- Systolic BP
- Diastolic BP
- Waist
- Hip

Feature selection



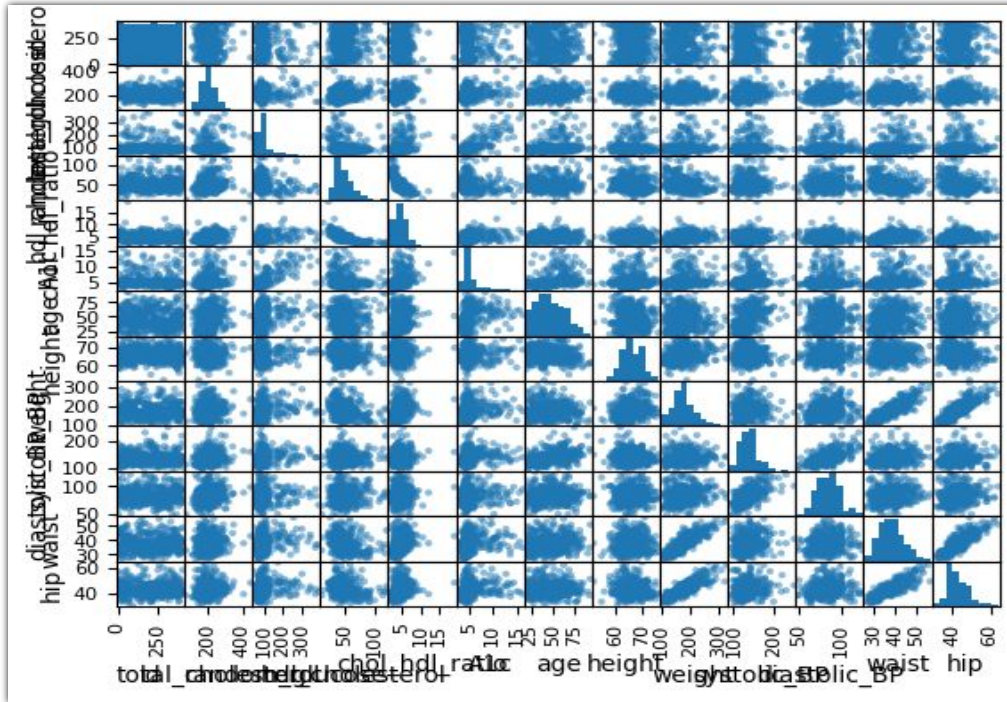
- Based on the scatter plot matrix, we have selected A1c, weight, and age features from the diabetes dataset.
- **Reasons why we chose:** The A1c is a key indicator for diagnosing diabetes and monitoring long-term blood sugar levels.
- Obesity(weight) is associated with insulin resistance and type-2 diabetes.
- Type 2 diabetes risk increases significantly as people age. This is because the body's ability to process insulin can decrease over time due to factors like lifestyle, metabolism, and genetic predisposition.

Data visualizations



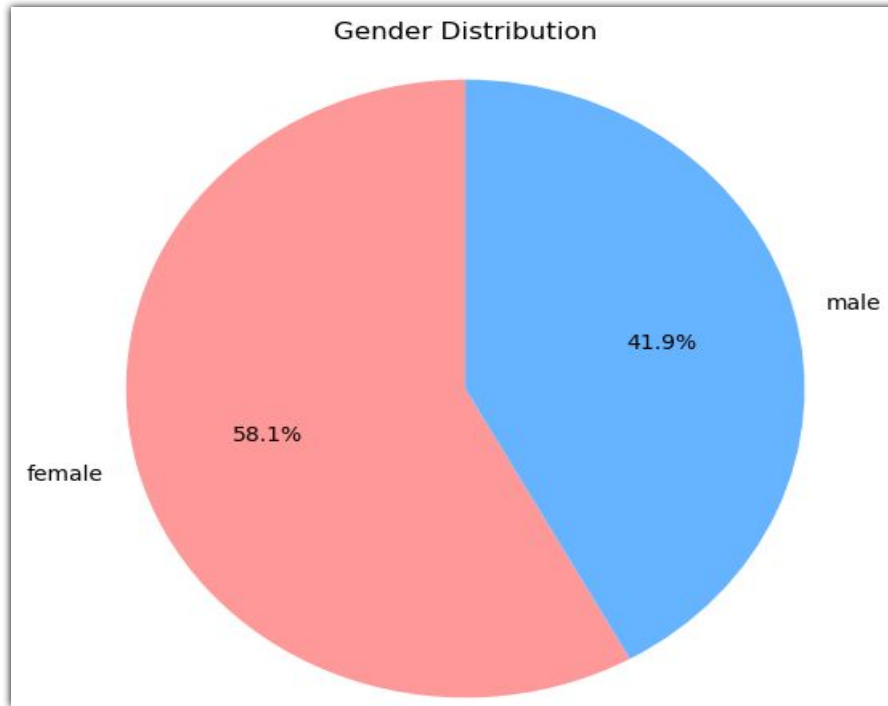
- Scatter plot for all features
- Pie chart showing the gender distribution
- Bar chart of A1c levels of age
- Histogram and boxplot of weight
- Histogram and boxplot of A1c

Scatter plot for all the features



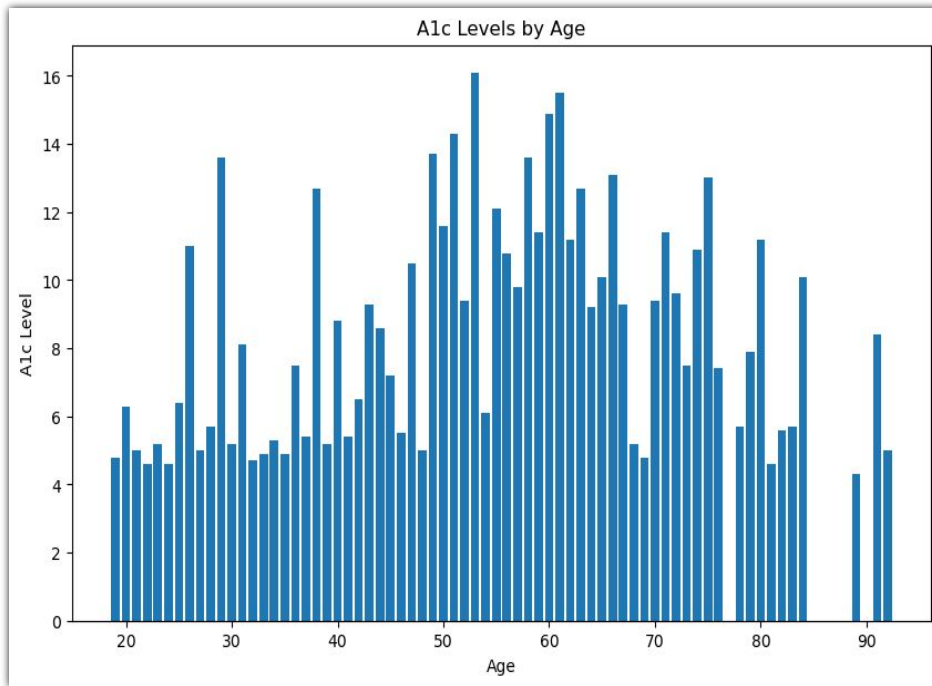
The scatter plot shows the strong positive correlation between the A1c and random glucose, weight and hip, waist and hip. The A1c is a feature that is considered as an important risk factor for diabetes along with weight which indirectly contribute to obesity, hence a reason they are selected as an important feature.

Pie chart showing gender distribution



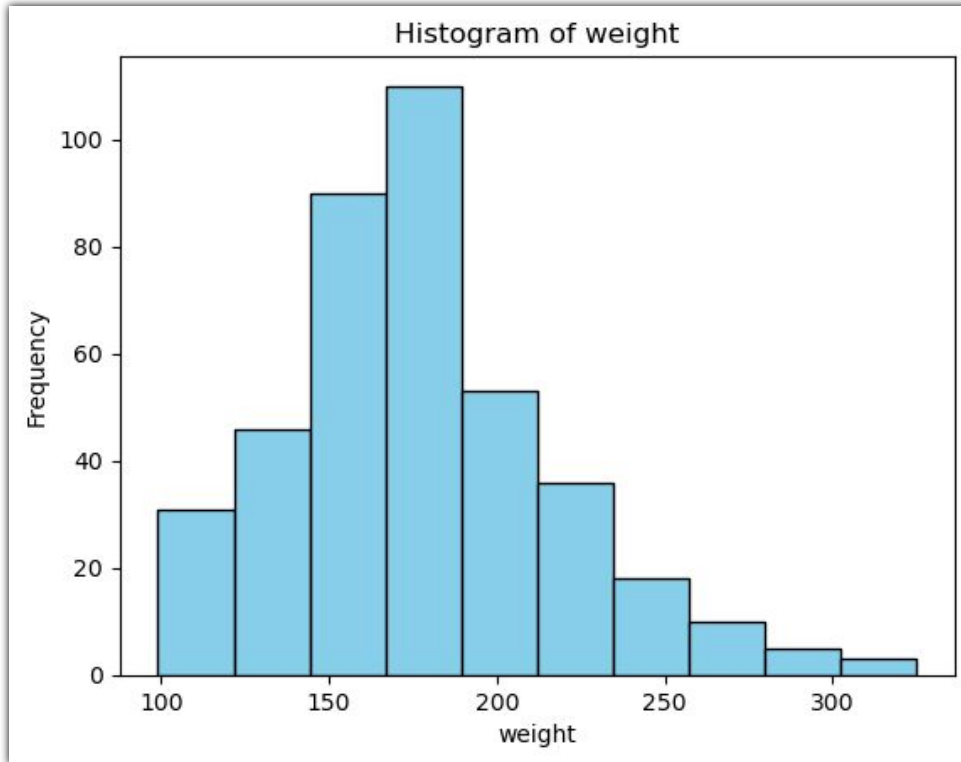
The distribution of individuals gender from the dataset shows 58.1% of females, that is slightly higher percentage of females in the study population compared to males who contribute to just 41.9%.

Bar chart of A1c levels by age



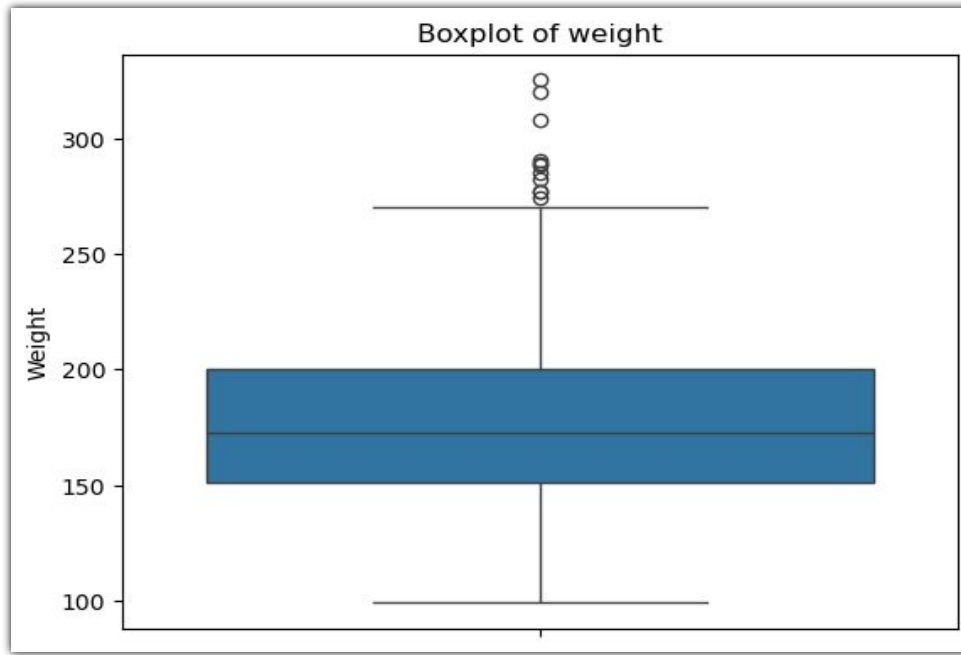
From the plot, Middle-aged individuals (50-70) have the highest risk of elevated A1c levels, suggesting a critical age range for the development of diabetes and pre-diabetes. A1c values vary significantly across age groups. While younger individuals (under 40) generally have lower A1c levels, some younger individuals still show elevated values, indicating that diabetes or pre-diabetes can occur at any age. After 70, some individuals show a decrease in A1c levels, though others still maintain high levels.

Histogram of weight



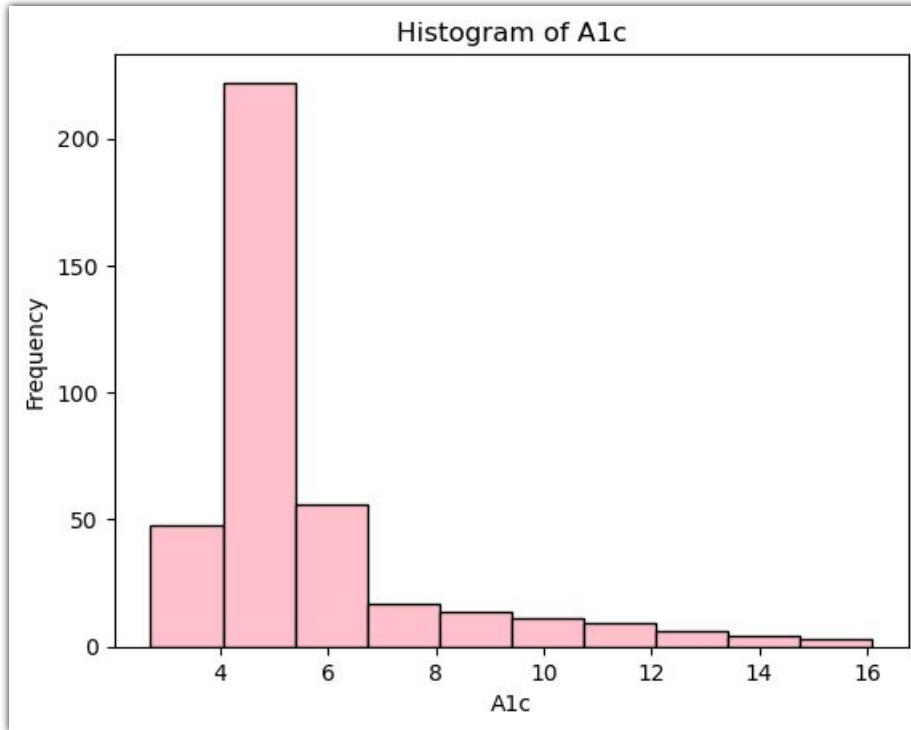
The histogram in the image represents the distribution of "weight" across different frequency ranges. The x-axis represents the weight values, while the y-axis shows the frequency of occurrences within each weight bin. The majority of the data is concentrated between the weights of 125 and 200, with a peak around 175. This indicates that most individuals (or data points) have weights in this range. As the weight increases beyond 180, the frequency drops, showing fewer occurrences for higher weight values.

Boxplot of weight



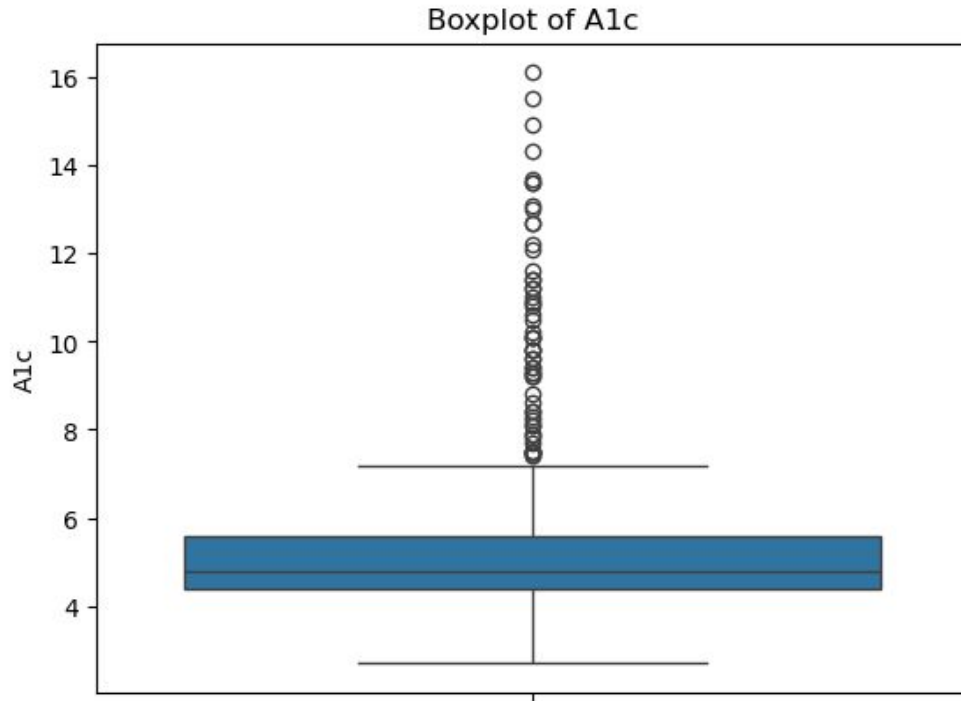
The boxplot shows the distribution of weight. The median weight is noted to be 175lbs, showing that half the data points fall above and half below this value. Above the whiskers at the level of 275, there are several dots that represent outliers that are individuals whose weights are unusually high compared to the rest of the dataset. These outliers indicate that a small number of individuals have significantly higher weights. Overall, the plot shows that most people fall within a relatively narrow weight range, but the presence of outliers highlights the variability in the dataset.

Histogram of A1c



The histogram shows the distribution of A1c levels. The distribution is heavily skewed to the right, with the majority of the A1c values clustered between 4 and 6, suggesting that most individuals have A1c levels in the normal range (typically below 5.7 for non-diabetics). As the A1c values increase, the frequency drops sharply, indicating fewer individuals with higher A1c levels, which may represent pre-diabetic or diabetic individuals (A1c levels above 6.5). The distribution after A1c levels of 8, showing only a small number of cases with values up to 16.

Boxplot of A1c



The box plot illustrates the distribution of A1c values. The interquartile range (IQR), shows the median A1c at 5. This shows that most individuals' A1c fall within a narrow range. The whiskers extending above 7.5 indicates the individuals with high A1c or possibly diabetes. Overall, the plot highlights a general concentration of A1c values, with some individuals having significantly elevated levels.

Confusion Matrix

```
import numpy as np
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report, confusion_matrix
data['diabetes'] = np.where(data['A1c'] > 7.0, 1, 0)
x = data[['age']]
y = data['diabetes']
# Define the model
model = LogisticRegression(solver='liblinear', C=10, random_state=0)
model.fit(x, y) #fit the model
# Evaluate the model
p_pred = model.predict_proba(x)
y_pred = model.predict(x)
scor = model.score(x, y)
conf_m = confusion_matrix(y, y_pred) # Calculate confusion matrix
print(scor)
print(conf_m)
```

```
0.843501326259947
```

```
[[317  2]
 [ 57  1]]
```

Interpretation: The confusion matrix results show that the logistic regression model, using age to predict whether A1c is greater than 7, has an overall accuracy of around 84.35%. The matrix indicates that out of 319 non-diabetic cases (class 0), the model correctly predicts 317, but misclassified 2 as diabetic (false positives). For diabetic cases (class 1), the model correctly identifies 1 out of 58 (true positives) while misclassifying 57 as non-diabetic (false negatives). This demonstrates that the model performs well in detecting non-diabetic cases but struggles significantly to identify diabetic cases. The low performance for class 1 might be due to class imbalance or insufficient feature information.

Accuracy

#Accuracy of model

```
print(classification_report(y,model.predict(x)))
```

	precision	recall	f1-score	support
0	0.85	0.99	0.91	319
1	0.33	0.02	0.03	58
accuracy			0.84	377
macro avg	0.59	0.51	0.47	377
weighted avg	0.77	0.84	0.78	377

Interpretation: The classification report reveals that the model performs well for class 0 but struggles with class 1. For class 0, the precision is 0.85, recall is 0.99, and the F1-score is 0.91, indicating effective identification of the majority class. However, for class 1, the model's performance is much weaker, with a precision of 0.33, recall of 0.02, and an F1-score of 0.03, suggesting difficulty in accurately predicting this class(class1). The overall accuracy is 84%, but this is largely influenced by the dominance of class 0. The macro average scores, which treat both classes equally, are much lower, with an F1-score of 0.47, reflecting the imbalance between the two classes. The support size indicates the dataset size which is medium in this case(377) indicating the good reliance on the dataset.

Linear Regression

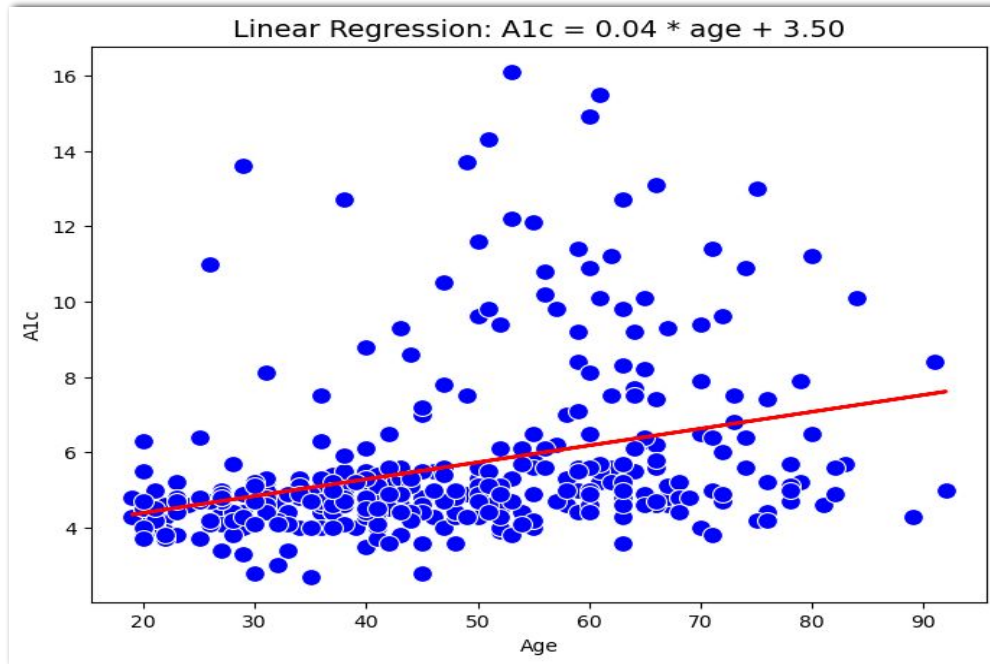


```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.linear_model import LinearRegression, LogisticRegression
X_linear = data[['age']]
y_linear = data['A1c']
linear_model = LinearRegression()
linear_model.fit(X_linear, y_linear)
slope = linear_model.coef_[0]
intercept = linear_model.intercept_
print(f"Linear Regression Equation: A1c = {slope:.2f} * age+ {intercept:.2f}")

Linear Regression Equation: A1c = 0.04 * age+ 3.50
```

Interpretation: The equation, $A1c = 0.04 \times \text{age} + 3.50$, indicates that for every 1-year increase in age, the A1c value is expected to increase by 0.05 units. The intercept of 3.50 represents the baseline A1c value when age is 0. While the slope is relatively small, the linear relationship suggests a modest positive correlation between age and A1c, indicating that older individuals are likely to have slightly higher A1c levels, which could be associated with elevated blood sugar levels over time.

Linear regression plot



Interpretation: The scatter plot illustrates the relationship between Age (on the x-axis) and A1c levels (on the y-axis). A linear regression line is plotted in red, showing the predicted A1c levels as a function of age. There's a lot of variability in A1c levels at any given age. The slope of the line (0.05) suggests that for every 1-year increase in age, the A1c level increases by 0.05 on average. The spread of the data points indicates that factors other than age contribute significantly to A1c levels, as people of the same age can have very different A1c values.

Elbow graph to spot the best 'K'

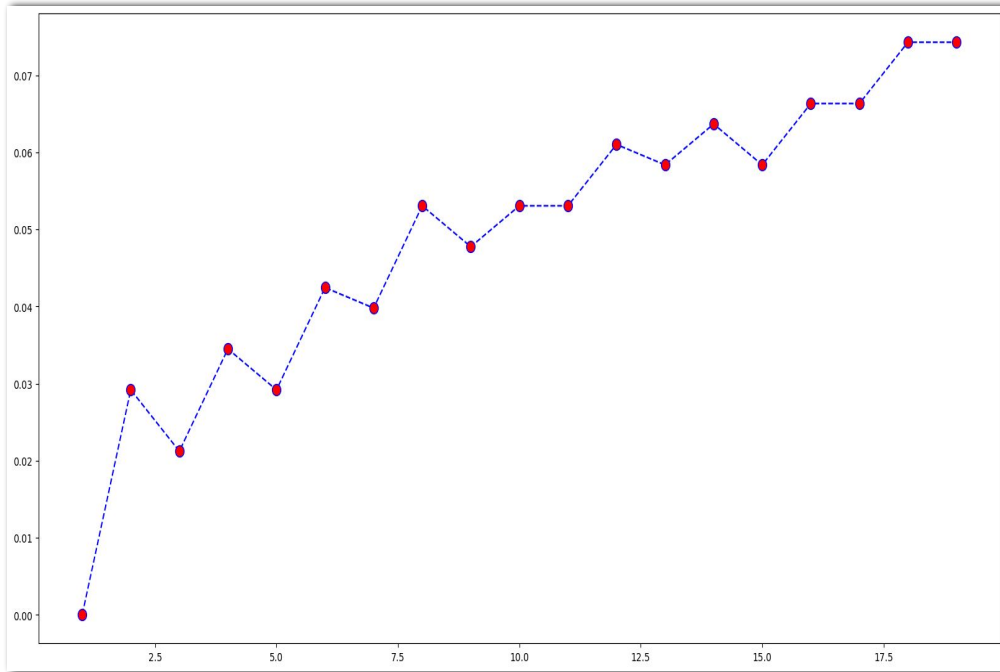


```
x=data[["A1c","age"]]
y=data["diabetes"]

import numpy as np
from sklearn.neighbors import KNeighborsClassifier
import matplotlib.pyplot as plt
SSE=[]
for k in range(1,20):
    knn=KNeighborsClassifier(n_neighbors=k)
    knn.fit(x,y)
    pred_i=knn.predict(x)
    SSE.append(np.mean(pred_i != y))
plt.figure(figsize=(20,10))
plt.plot(range(1,20),SSE,color='b',linestyle='dashed',marker='o',markerfacecolor='r',markersize=10)
plt.show()
```

We have chosen A1c and age as the features that have relation to the target outcome variable “diabetes”.

The picture indicates the code to plot a elbow graph to get the K-Nearest Neighbors.

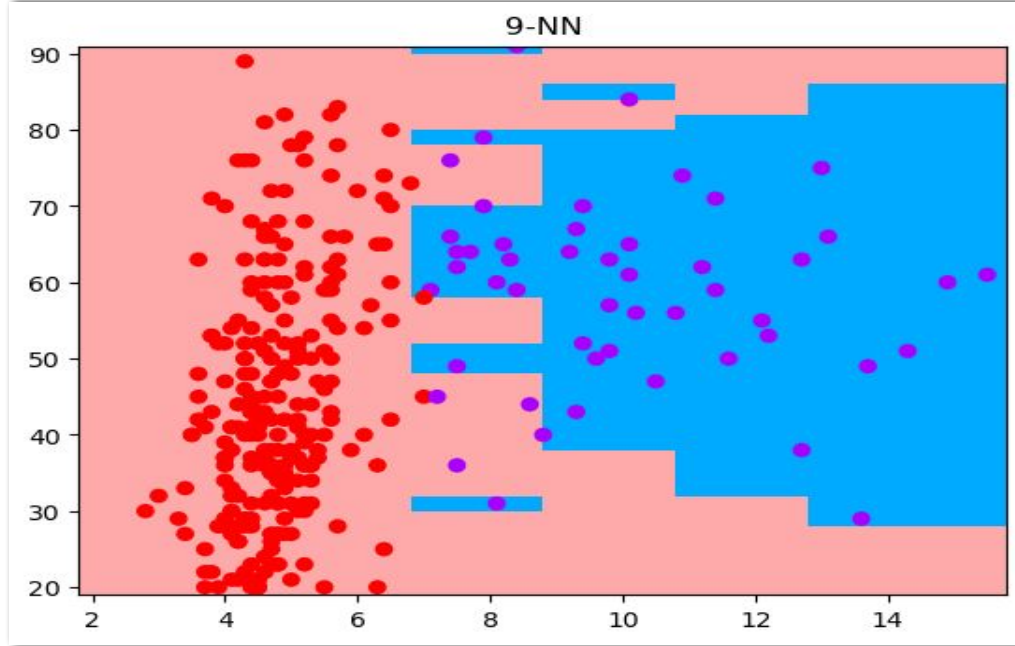


The best K-Nearest Neighbor is found to be 9 from the plot because there is slight modifications after this point, whereas from the origin, until 9 nearest neighbors, the graph has shown steeper fluctuations.

K-Nearest Neighbors

```
: from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(x, y, test_size = 0.2)

: from matplotlib.colors import ListedColormap
n_neighbors=9
X=X_train.values[:,0:2]
h=2
clf=KNeighborsClassifier(n_neighbors,weights='distance')
clf.fit(X,y_train)
x_min, x_max=X[:,0].min()-1, X[:,0].max()+1 # x axis A1c
y_min, y_max=X[:,1].min()-1, X[:,1].max()+1 # y axis age
xx,yy=np.meshgrid(np.arange(x_min,x_max,h),np.arange(y_min,y_max,h)) # grids
cmap_l =ListedColormap(['#FFAAAA', '#AAFFAA', '#00AAFF']) # predicted
cmap_d =ListedColormap(['#FF0000', '#00FF00', '#AA00FF']) # actual
Z= clf.predict(np.c_[xx.ravel(),yy.ravel()])
Z=Z.reshape(xx.shape)
plt.figure()
plt.pcolormesh(xx,yy,Z,cmap=cmap_l) # predicted values on back
plt.scatter(X[:,0],X[:,1],c=y_train,cmap=cmap_d) # actual values
plt.xlim(xx.min(),xx.max())
plt.ylim(yy.min(),yy.max())
plt.title("9-MN")
plt.show()
```



The background color represents the predicted class in each region of the plot. Red and purple points represent actual data points from non-diabetic and diabetic respectively. The decision boundary is the jagged line that separates the pink and blue regions. It uses 9 neighbors to make predictions, the decision boundary is not smooth but is heavily influenced by local clusters of training data. The left side of the plot (red region) has a densely packed cluster of red points, and the classifier predicts this area as belonging to non-diabetic. Whereas, blue region of the plot contains more purple points spread distantly, so the classifier predicts this area as diabetic. There are few purple points that fall in the pink region, so these areas are where the classifier was unable to perfectly distinguish between the two classes due to overlapping regions in the dataset.

KNN classifier accuracy



```
train_accuracy = clf.score(X_train, y_train)
test_accuracy = clf.score(X_test, y_test) |
print(train_accuracy)
print(test_accuracy)
```

```
1.0
0.9605263157894737
```

The training accuracy using the K-Nearest Neighbor classifier for the diabetes data is found to be 100% while the test accuracy is found to be only 96.05%, which indicates that the model perfectly classifies all training data points. The test data accuracy indicates overfitting, which doesn't generalize effectively to test data.

Naive-Bayes classifier

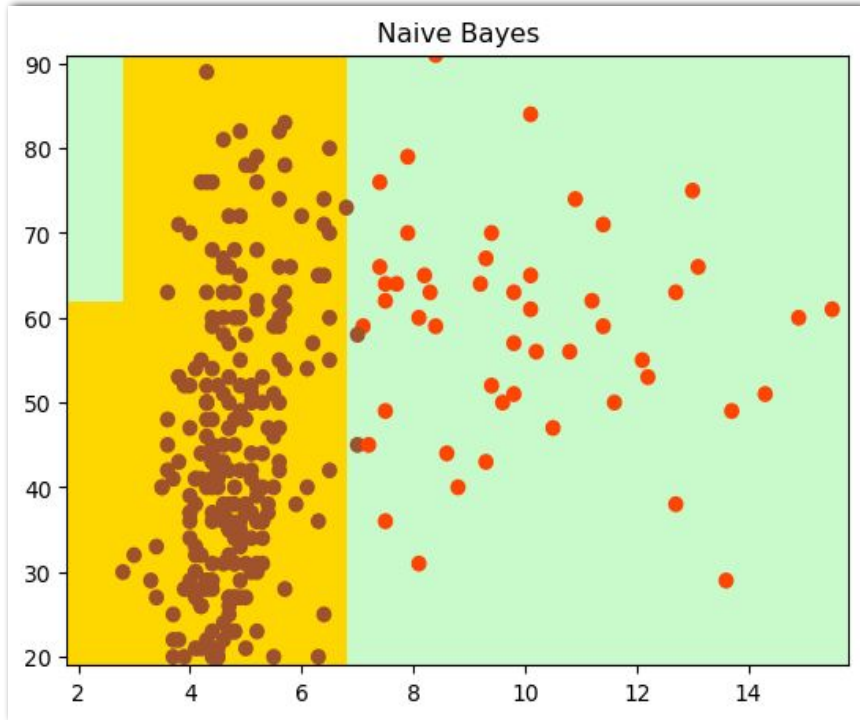
Naive Bayes

```
from sklearn.naive_bayes import GaussianNB
gclf = GaussianNB()
gclf.fit(X_train,y_train)
```

▼ GaussianNB 1 2

GaussianNB()

```
x_min, x_max=X[:,0].min()-1, X[:,0].max()+1 # x axis Alc
y_min, y_max=X[:,1].min()-1, X[:,1].max()+1 # y axis age
xx,yy=np.meshgrid(np.arange(x_min,x_max,h),np.arange(y_min,y_max,h)) # grids
cmap_1 =ListedColormap(['#FFD700', '#B2EBF2', '#C8FACC']) # predicted
cmap_d =ListedColormap(['#A0522D', '#FFD700', '#FF4500']) # actual
Z= gclf.predict(np.c_[xx.ravel(),yy.ravel()])
Z=Z.reshape(xx.shape)
plt.figure()
plt.pcolormesh(xx,yy,Z,cmap=cmap_1) # predicted values on back
plt.scatter(X[:,0],X[:,1],c=y_train,cmap=cmap_d) # actual values
plt.xlim(xx.min(),xx.max())
plt.ylim(yy.min(),yy.max())
plt.title("Naive Bayes")
plt.show()
```

The background color represents the predicted class in each region of the plot. Orange and brown points represent actual data points from non-diabetic and diabetic, respectively. The decision boundary is the jagged line separating the yellow and light-green regions. In the yellow region on the left, the classifier predicts most points as belonging to non-diabetic, where brown points (diabetic) are densely clustered but intermixed with orange points from non-diabetic. However, the majority of the region is predicted as non-diabetic, likely due to the overlap and density of these points. The light-green region on the right is predicted as diabetic, containing more scattered orange points. These areas represent where the classifier identified the presence of diabetic despite occasional points from non-diabetic falling in this space.

Naive Bayes classifier accuracy



```
train_accuracy=clf.score(X_train, y_train)
test_accuracy=clf.score(X_test, y_test)
print(train_accuracy)
print(test_accuracy)
```

```
0.9900332225913622
0.9868421052631579
```

The training accuracy using the Naive Bayes classifier for the diabetes data is found to be 99% while the test accuracy is found to be only 98.68%, showing it not only learns the training data well but also generalizes effectively to new, unseen data, making it reliable for predicting diabetes. These accuracies mean it's a better choice when compared to the KNN classifier.

The best model for the diabetes data based on accuracy among KNN and NB



- The best classifier model among K-Nearest Neighbors (KNN), and Naive Bayes (NB) classifiers for the diabetes data is found to be Naive Bayes classifier.
- Because the training and test accuracies for KNN is found to be 100 and 96.05% respectively, indicating that the model doesn't effectively generalize to new test data whereas the training and test accuracies of Naive Bayes is found to be 99 and 98.68% respectively, despite a low training accuracy, NB suggests good generalization ability with a slight difference in train and test accuracy.
- It is clear that, Naive Bayes classifier is the well-performing model that balances complexity and simplicity for the diabetes data.

Support Vector Machine (SVM)

```
Support Vector Machine Models

[30]: Xdata[["A1c", "age"]]
      Ydata["diabetes"]
      xx=X.iloc[:100,:]
      yy=Y.iloc[:100]

[31]: # define- fit - predict - find the accuracy of models
      from sklearn import svm
      from sklearn.metrics import accuracy_score
      # Linear kernel
      svc=svm.SVC(kernel='linear', C=1).fit(x,y)
      svc_pred=svc.predict(x)
      print('svc accuracy',accuracy_score(y,svc_pred)*100)
      # Linear svm
      lin_svc=svm.LinearSVC(C=1).fit(x,y)
      lsvc_pred=lin_svc.predict(x)
      print('linear svc accuracy',accuracy_score(y,lsvc_pred)*100)
      # rbf kernel
      rbf_svc=svm.SVC(kernel='rbf', gamma=0.7, C=1).fit(x,y)
      rsvc_pred=rbf_svc.predict(x)
      print('rbf accuracy',accuracy_score(y,rsvc_pred)*100)
      # poly kernel, degree 3  $ax^3+bx^2+cx+d$ 
      poly_svc=svm.SVC(kernel='poly', degree=3, C=1).fit(x,y)
      psvc_pred=poly_svc.predict(x)
      print('poly accuracy',accuracy_score(y,psvc_pred)*100)

svc accuracy 100.0
linear svc accuracy 96.0
rbf accuracy 100.0
poly accuracy 98.0
```

Interpretation: The SVC model achieved perfect accuracy with 100%, classifying all samples correctly. This suggests that this kernel function is well-suited to the data and captures the patterns fully. The Linear SVC achieved 96% accuracy. While this is high, it's slightly lower than the SVC model with 100% accuracy, suggesting that a linear decision boundary may not fully capture all patterns in the data. The SVC with a RBF kernel achieved perfect accuracy with 100%. The RBF kernel, which can model more complex decision boundaries, is likely highly effective on this dataset. The polynomial kernel achieved 98% accuracy, which is also very high but slightly less than the RBF and the default SVC models. This suggests it can model some non-linear patterns but may not be as well-suited to this specific data as the RBF kernel.

Code to plot the SVM with 4 kernels

```
import numpy as np
import matplotlib.pyplot as plt

h = 0.02
# Setting up the meshgrid for plotting decision boundaries
x_min, x_max = x.iloc[:, 0].min() - 1, x.iloc[:, 0].max() + 1 # A1c over x axis
y_min, y_max = x.iloc[:, 1].min() - 1, x.iloc[:, 1].max() + 1 # age over y axis
xx, yy = np.meshgrid(np.arange(x_min, x_max, h), np.arange(y_min, y_max, h))

titles = ['linear kernel', 'linear svc', 'rbf', 'poly']

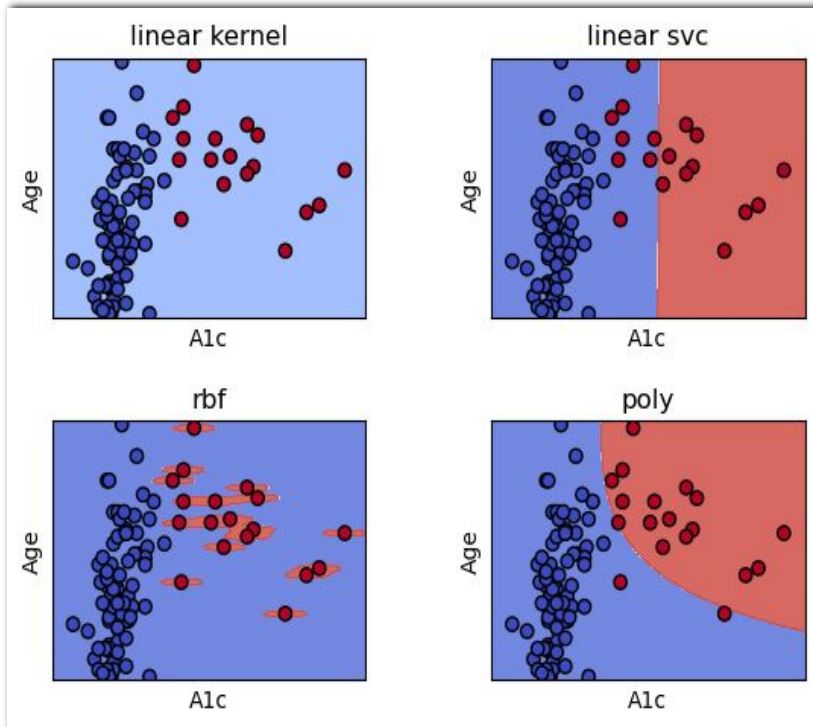
# Loop through each classifier
for i, clf in enumerate((svc, lin_svc, rbf_svc, poly_svc)):
    plt.subplot(2, 2, i + 1)
    plt.subplots_adjust(wspace=0.4, hspace=0.4)

    # Predict the classifier boundaries on the meshgrid
    Z = clf.predict(np.c_[xx.ravel(), yy.ravel()])
    Z = Z.reshape(xx.shape)

    # Plot decision boundaries and data points
    plt.contourf(xx, yy, Z, cmap=plt.cm.coolwarm, alpha=0.8)
    plt.scatter(x.iloc[:, 0], x.iloc[:, 1], c=y, cmap=plt.cm.coolwarm, edgecolor='k')

    # Axis labels and title
    plt.xlabel('A1c')
    plt.ylabel('Age')
    plt.xlim(xx.min(), xx.max())
    plt.ylim(yy.min(), yy.max())
    plt.xticks(())
    plt.yticks(())
    plt.title(titles[i])

plt.show()
```



Interpretation: This image shows the decision boundaries of four different SVM classifiers applied to diabetes dataset with two features, "A1c" and "Age".

- This SVM model uses a linear kernel, which does not display any decision boundary, indicating that the linear kernel SVM was unable to separate the classes in this feature space. This likely means that a linear kernel isn't suited for diabetes dataset, as the data points are not linearly separable based on "A1c" and "Age."
- The linear SVC plot shows a linear decision boundary, but it doesn't capture all of the nuances in the data. The straight-line boundary results in several misclassified points, as the linear SVC is only effective for datasets with a clear linear separation, which isn't the case here.



Interpretation continued..

- The RBF kernel creates a complex, non-linear boundary that wraps around clusters of red and blue points, following the distribution closely and resulting in few misclassifications. This kernel effectively handles non-linear data and performs well for this dataset.
- The polynomial kernel also produces a non-linear boundary with some curvature, but it's less flexible than the RBF boundary. It captures the non-linear patterns somewhat but still miss classifies a few points, as its boundary doesn't adapt as precisely to the data.

Random Forest Classifier

Random forest classifier

```
# import packages
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn import datasets
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier

X=data[["A1c","age","weight"]]
y=data["diabetes"]
# split data to train and test
X_train, X_test, y_train, y_test=train_test_split(X,y, test_size=0.3 )
# define the model, fit the model
clf=RandomForestClassifier(n_estimators=50)
clf.fit(X_train, y_train)
RandomForestClassifier(n_estimators=50)
y_pred=clf.predict(X_test)

from sklearn.metrics import confusion_matrix, accuracy_score, classification_report
CM=confusion_matrix(y_pred,y_test)
print(CM)
AS=accuracy_score(y_pred,y_test)
print(AS)
CR=classification_report(y_pred,y_test)
print(CR)
```

```
[[96  1]
 [ 0 17]]
0.9912280701754386
```

	precision	recall	f1-score	support
0	1.00	0.99	0.99	97
1	0.94	1.00	0.97	17
accuracy			0.99	114
macro avg	0.97	0.99	0.98	114
weighted avg	0.99	0.99	0.99	114

```
rf=RandomForestClassifier(n_estimators=1)
rf.fit(X_train,y_train)
```

RandomForestClassifier

RandomForestClassifier(n_estimators=1)

```
len(rf.estimators_)
```

```
1
```

```
from sklearn import tree
X=data[["A1c","age","weight"]]
y=data["diabetes"]
plt.figure(figsize=(20,20))
_ = tree.plot_tree(rf.estimators_[0], filled=True)
```

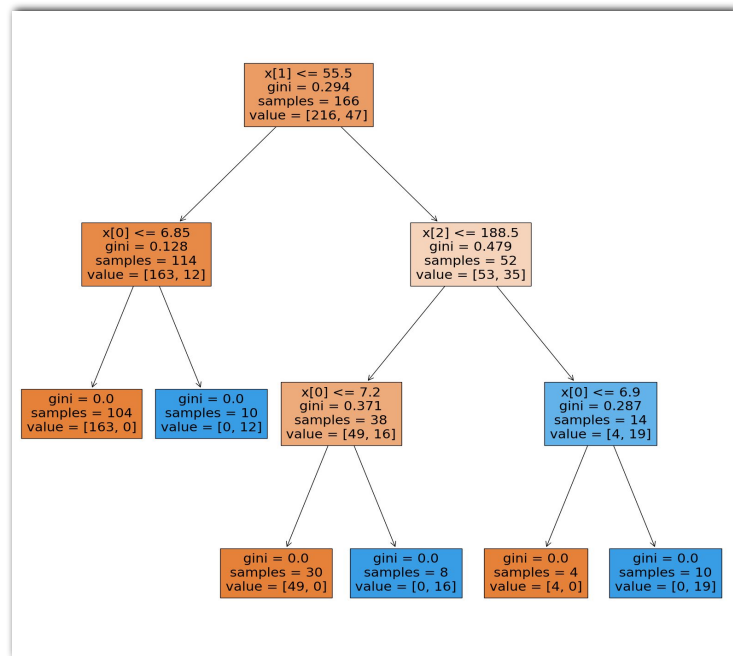

Accuracy of random forest classifier



- According to the confusion matrix, the model correctly identified 96 non-diabetic individuals(True negatives) and 17 diabetic individuals(True positives). There was only 1 false negative that is an actual diabetic case predicted as non- diabetic and no false positives.
- The accuracy score using the random forest classifier model is found to be 99.12%, which suggests that the model is highly effective at predicting whether an individual has diabetes or not, with a rate of 99.12% accuracy.
- According to the classification report, the precision for class 0 (non-diabetic) is 1.00, that is all predictions for the non-diabetic cases were correct, with zero false positives. Recall for class 1(diabetic) is 1.00, meaning all diabetic cases were identified correctly. The F1 score for both the classes are also high with 0.99 for non-diabetic and 0.97 for diabetic, which indicates that a good balance of precision and recall for both classes, with slightly more variation in identifying diabetic cases.
- Random forest classifier worked highly effective for the diabetes dataset.

Interpretation of random forest classifier


The plot represents a random forest graph that splits data points at different levels based on conditions applied to features (A1c, Age, and Weight). Each node shows a decision split, where the tree chooses a threshold on a specific feature to separate the data. The root node split in the tree, chosen based on the age feature (denoted as $x[1]$). This suggests that age is an influential factor in predicting diabetes. The threshold is $\text{age}=55.5$, with a Gini impurity of 0.294. The leaf nodes represent final predictions made by the tree. For example, one leaf node has $\text{gini} = 0.0$, $\text{samples} = 10$, and $\text{value} = [0, 12]$, indicating that all samples in this node are diabetic cases. This decision tree plot illustrates how the Random Forest model separates individuals based on age, A1c, and weight, identifying regions in the feature space that are more likely to contain diabetic or non-diabetic cases. The purity of many nodes ($\text{Gini} = 0.0$) indicates the tree has a strong discriminative ability.



Conclusion



- The random forest classifier is the most effective method for our diabetes dataset due to its high accuracy, excellent class balance, and superior performance in predicting both diabetic and non-diabetic classes. Random forest classifier performed the best with an accuracy of 99.12% and excellent metrics for precision, recall, and F1 scores for both diabetic and non-diabetic classes, making it reliable and robust.
- Naive Bayes classifier also performed well, with a test accuracy of 98.68%. Its generalization capability is strong, suggesting that it is well-suited for this dataset. However, its accuracy is slightly lower compared to the random forest model, and it doesn't handle class imbalance effectively.
- Support Vector Machine with RBF kernel achieved 100% accuracy and performs very well on this dataset. While the accuracy is perfect, the model's potential overfitting to training data may make it less practical compared to Random forest, which balances simplicity, interpretability, and accuracy.

- 
- KNN exhibits overfitting, with 100% training accuracy but slightly lower test accuracy(96.05%). The gap indicates it may not generalize as effectively to new data, making it less favorable compared to Random forest or Naive Bayes.
 - Logistic regression has a low accuracy in identifying diabetic cases, likely due to class imbalance or limited features. This makes it unsuitable as a primary choice for this dataset.
 - Linear SVC and polynomial kernel SVC achieve high accuracies(96% and 98% respectively) but are still slightly outperformed by the Random forest classifier in terms of overall effectiveness and balance between precision and recall.

References



1. Gupta NP. Diabetes prediction using machine learning. *Deleted Journal*. 2024;20(7s):2244-2257.
doi:10.52783/jes.3960
2. Kaur H, Kumari V. Predictive modelling and analytics for diabetes using a machine learning approach. *Applied Computing and Informatics*. Published online December 2018.
doi:<https://doi.org/10.1016/j.aci.2018.12.004>

Contributions



Project 1: Sai Teja Gajji, Reshmitha Mandava

Project 2: Pravanith Reddy, Haris Chandra Naik Gugulothu

Project 3: Venkata Sai Pallavi Pallapolu, Supraja Medicherla

Project 4: Sai Teja Gajji, Haris Chandra Naik Gugulothu

Project 5: Reshmitha Mandava, Pravanith Reddy

Project 6: Venkata Sai Pallavi Pallapolu, Supraja Medicherla

Coding: Venkata Sai Pallavi Pallapolu, Supraja Medicherla