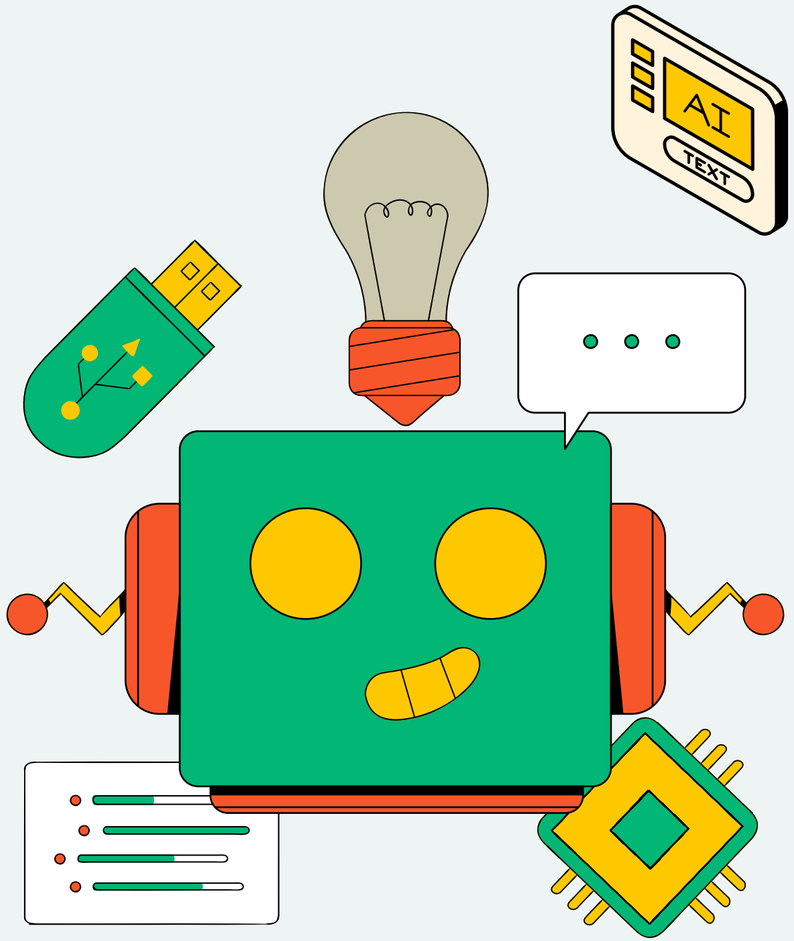
WE LEARN FOR THE FUTURE

DISEASE PREDICTION: LEVERAGING SUPPORT VECTOR MACHINES (SVM) AND K-NEAREST NEIGHBORS (KNN)"



PRESENTED BY:

- Giri Chandragiri
- Shaagun Suresh
- Sravanthi Kadari

PRESENTATION OUTLINE

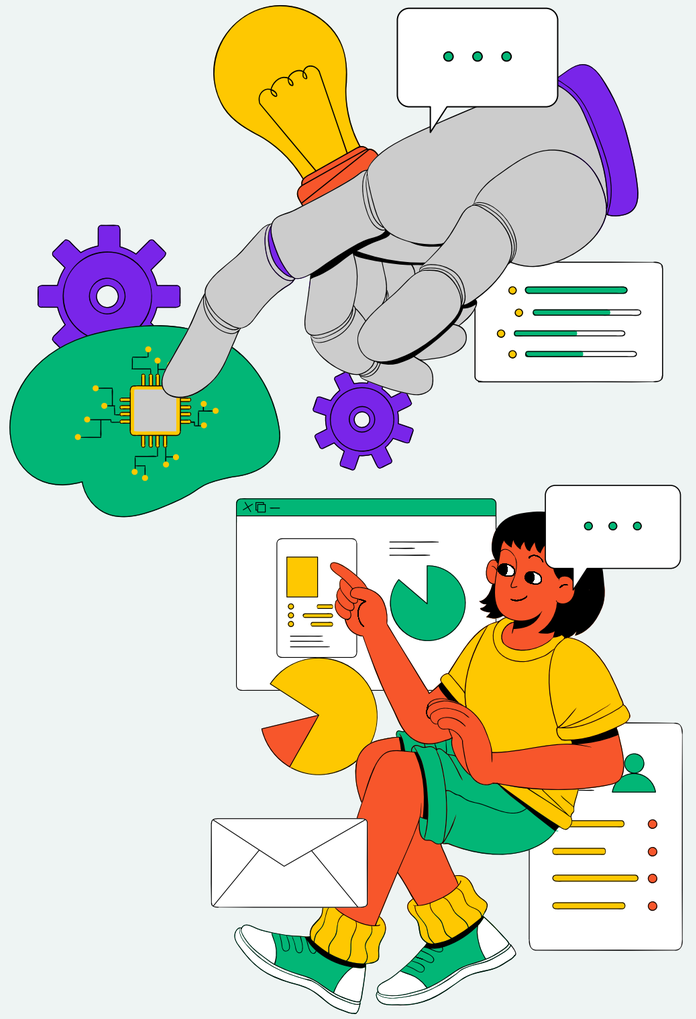
- Introduction
- Dataset Overview
- Data Preprocessing
- Model Selection
- Model Evaluation
- Fine – Tuning
- Model Testing
- Conclusion

INTRODUCTION

- Disease Symptom and Patient Profile Dataset is taken from Kaggle
- Importance of medical research and healthcare analytics
- Purpose of the presentation: Exploring Support Vector Machines (SVM) and k-Nearest Neighbors (kNN) algorithms for disease prediction

DATASET OVERVIEW

Disease: Names of diseases for analysis and comparison.
Symptoms: Fever, Cough, Fatigue, Difficulty Breathing.
Demographics: Age (continuous), Gender (categorical).
Health Indicators: Blood Pressure, Cholesterol Level.
Outcome Variable: Binary result of diagnostic assessment.



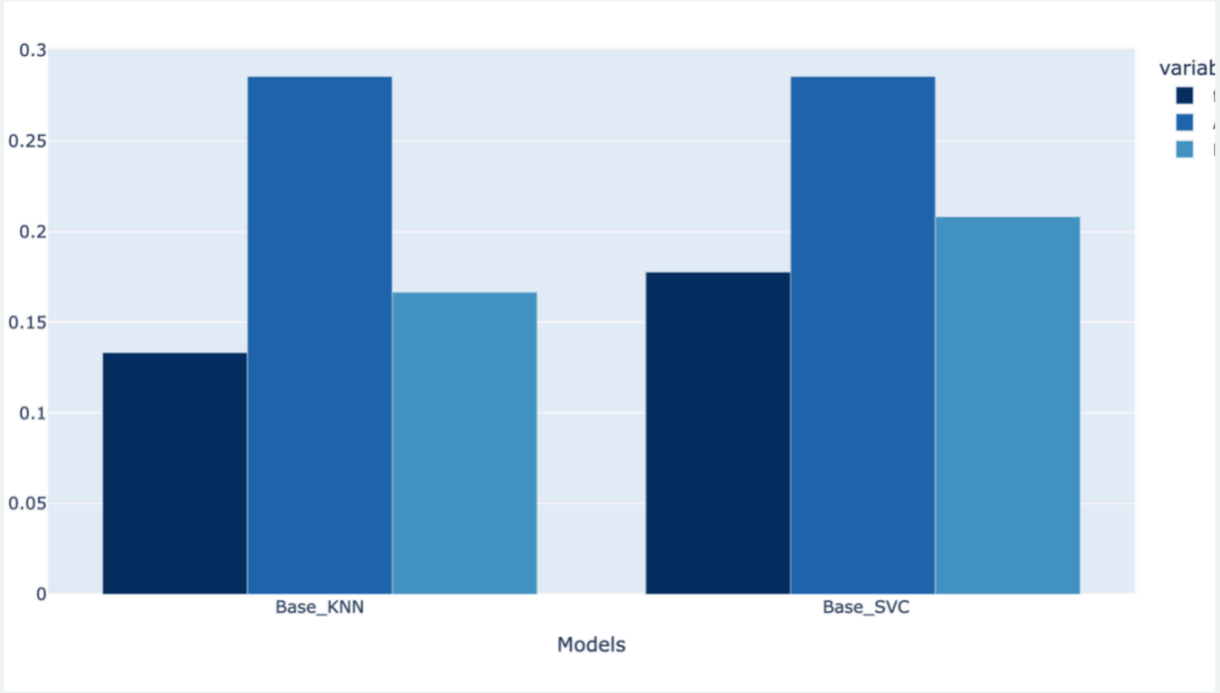
DATA PREPROCESSING

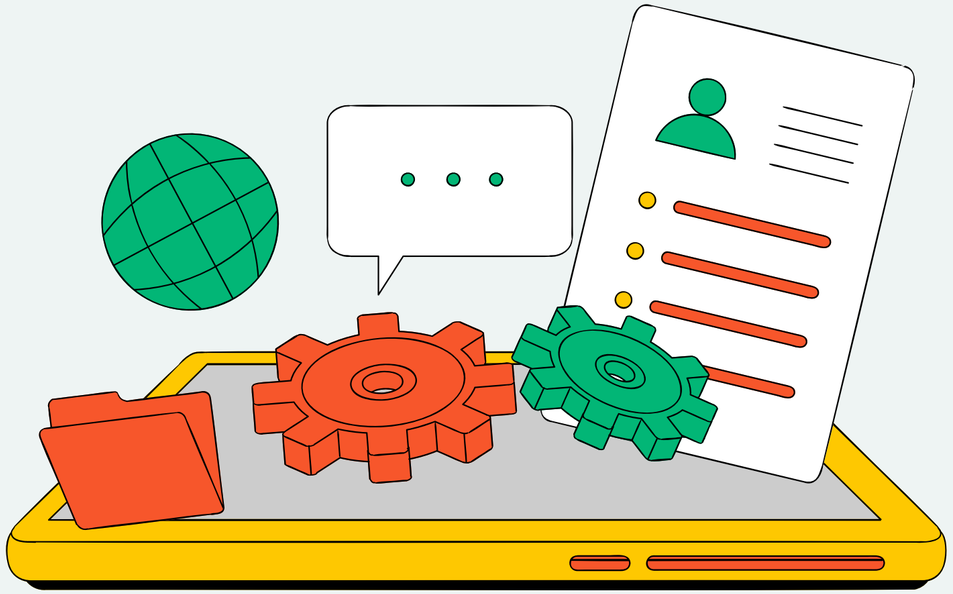
- The duplicated values are being removed.
- Upon examining the 'Disease' column, large number of unique diseases are present , many of which have only 1 to 5 samples. For a reliable disease prediction model, this sample size is insufficient.

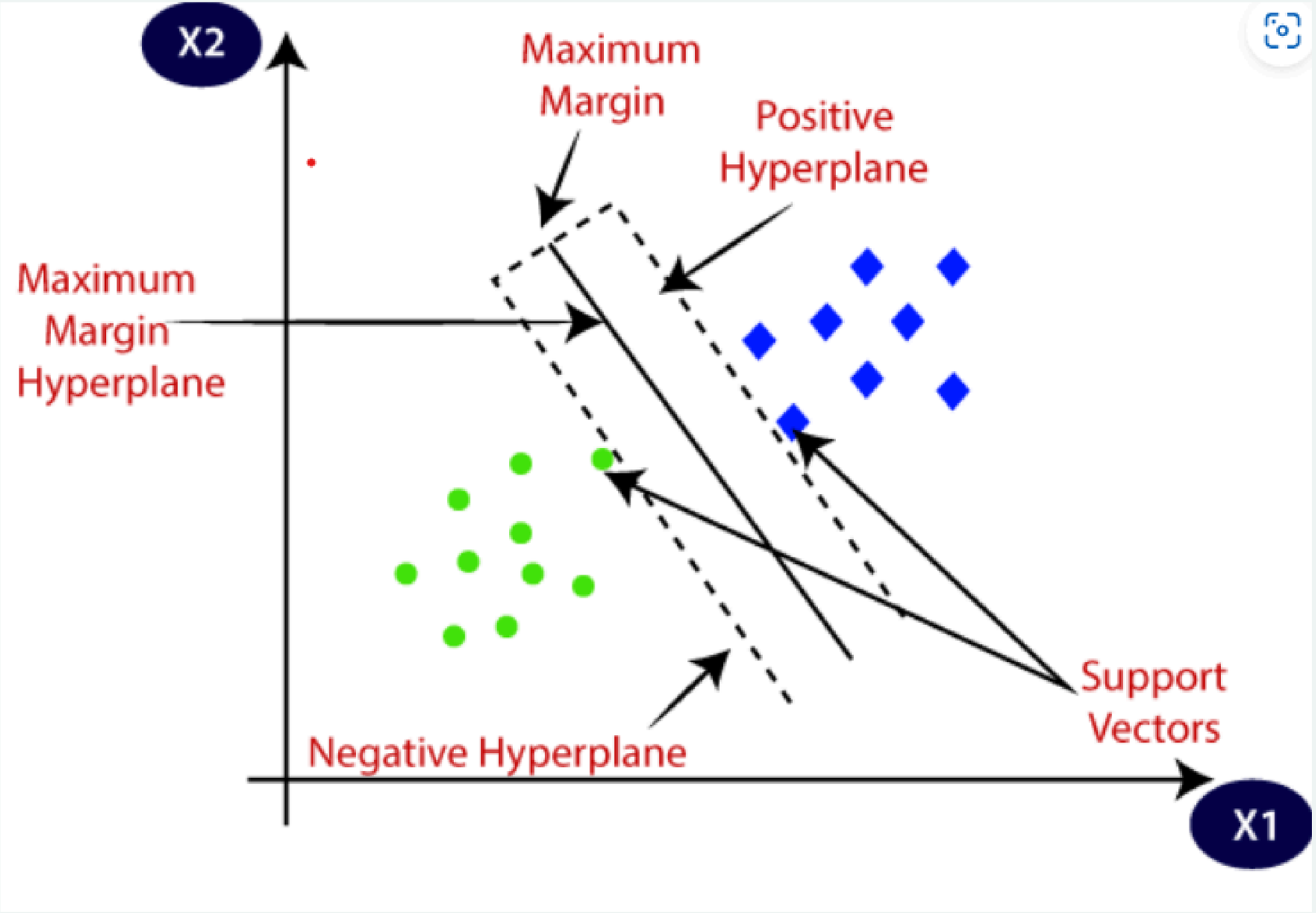
MODEL SELECTION

After training our K-Nearest Neighbors (K-NN) and Support Vector Machines (SVM) models, we observe that SVM significantly outperforms K-NN in terms of the macro-averaged F1 score.

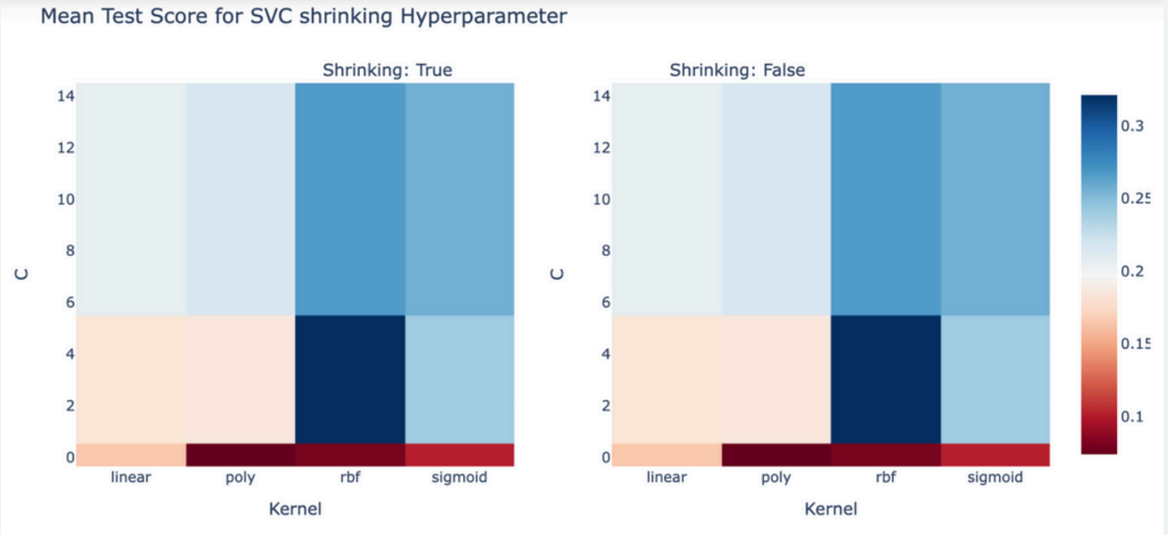
Given this performance difference, it makes sense to focus our efforts on the SVM model. We will proceed with fine-tuning this model to see if we can further improve the F1 score






X2
Maximum
Margin
Positive
Hyperplane
Maximum
Margin
Hyperplane
Support
Vectors
Negative Hyperplane
X1

FINE - TUNING



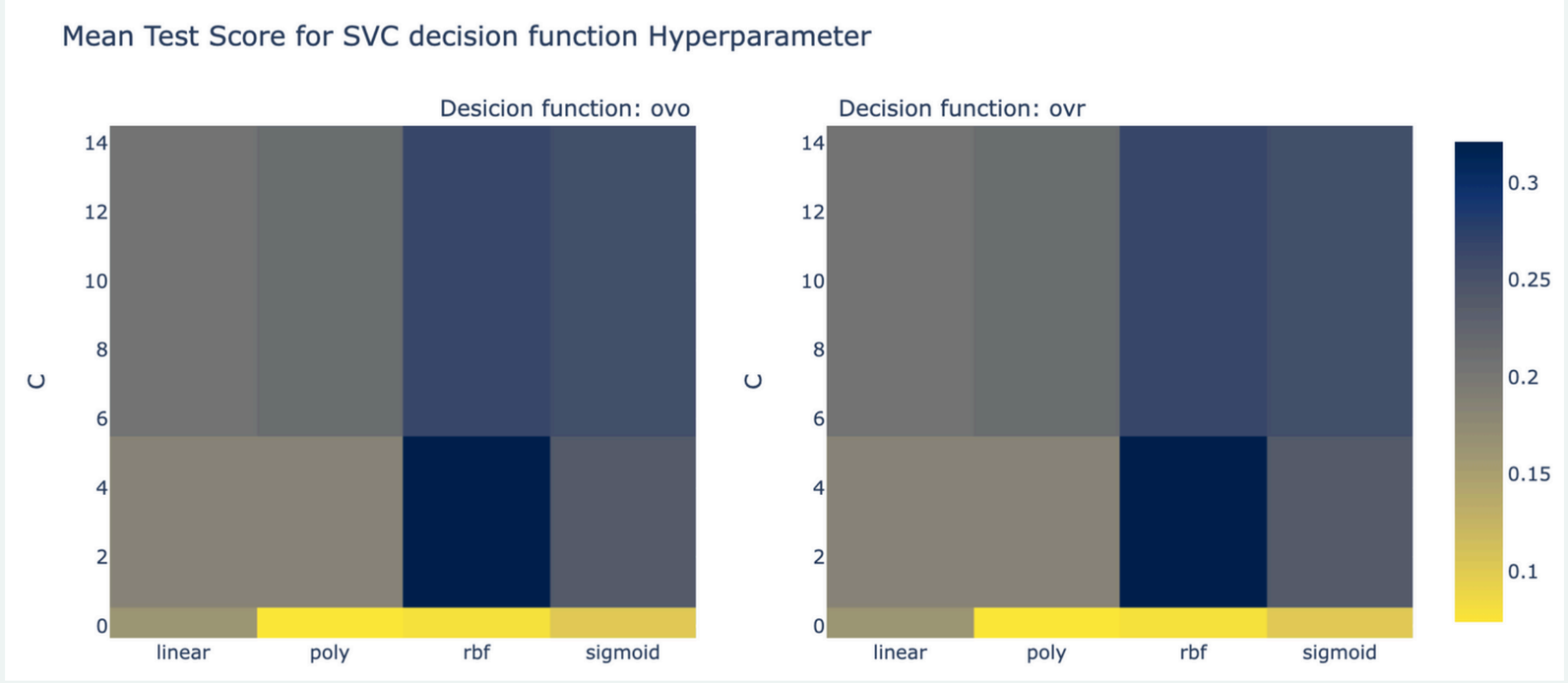
Objective: Enhance model performance and predictive accuracy.

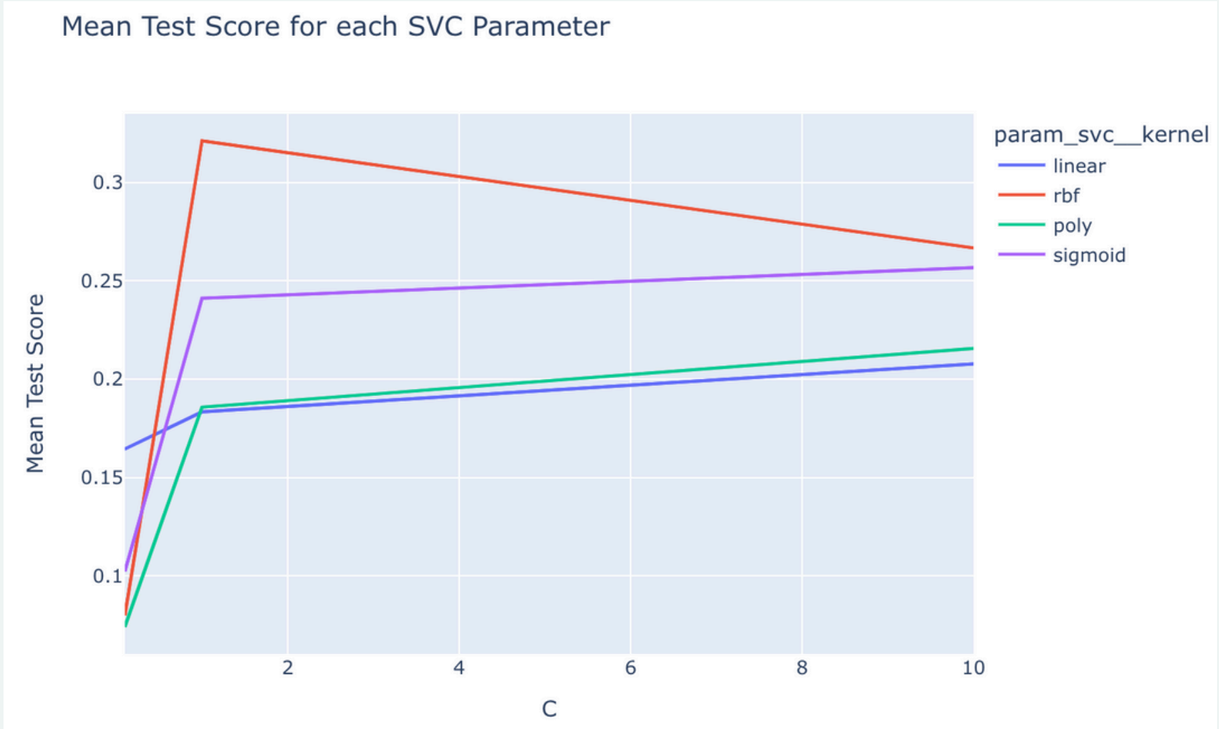
Hyperparameters: Control algorithm behavior and pattern capture.

Techniques: Grid search, random search, Bayesian optimization.

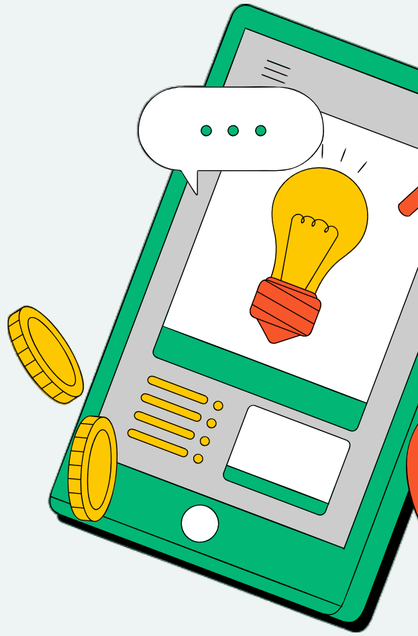
Parameters: Regularization strength, kernel type, distance metric, etc.

Purpose: Improve classification accuracy, reduce bias/variance.


Mean Test Score for SVC decision function Hyperparameter
Desicion function: ovo
Decision function: ovr
C
14
12
10
8
6
4
2
0
linear
poly
rbf
sigmoid
0.3
0.25
0.2
0.15
0.1


Mean Test Score for each SVC Parameter
Mean Test Score
0.3
0.25
0.2
0.15
0.1
2
4
6
8
10
C
param_svc__kernel
linear
rbf
poly
sigmoid

KNN(K-NEAREST NEIGHBORS)



The k represents the number of nearest neighbors to consider when making predictions. It is a hyperparameter that needs to be specified by the user. Choosing the right value of k is crucial, as it can significantly impact the performance of the algorithm

Model Selection

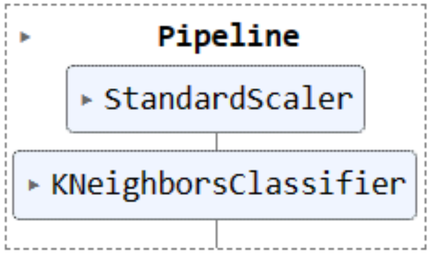
In [89]:
```
X = df.drop(['Disease'], axis= 1).values
y = df.Disease.values
```

In [90]:
```
X_train, X_val, y_train, y_val = train_test_split(X, y, test_size= 0.4, shuffle= True, stratify= y, random_state=30)
X_val, X_test, y_val, y_test = train_test_split(X_val, y_val, test_size= 0.5, shuffle= True, stratify= y_val, random_stat
```

In [91]:
```
svc_pipe = Pipeline([('scaler', StandardScaler()), ('svc', SVC(class_weight= 'balanced'))])
knn_pipe = Pipeline([('scaler', StandardScaler()), ('knn', KNeighborsClassifier())])
```

In [92]:
```
svc_pipe.fit(X_train, y_train)
knn_pipe.fit(X_train, y_train)
```

Out[92]:



In [93]:
```
ysvc_pred = svc_pipe.predict(X_val)
yknn_pred = knn_pipe.predict(X_val)
```

Given our problem of multi-class classification with imbalanced classes, the F1 score (macro-averaged) is an appropriate choice. The F1 score is the harmonic mean of precision and recall, and it gives a better measure of the incorrectly classified cases than the accuracy metric.

The macro-averaged F1 score calculates the F1 score for each class independently and then takes the average. This treats all classes equally, regardless of their imbalance, which is exactly what we need for our problem.

Our goal is to maximize this F1 score.

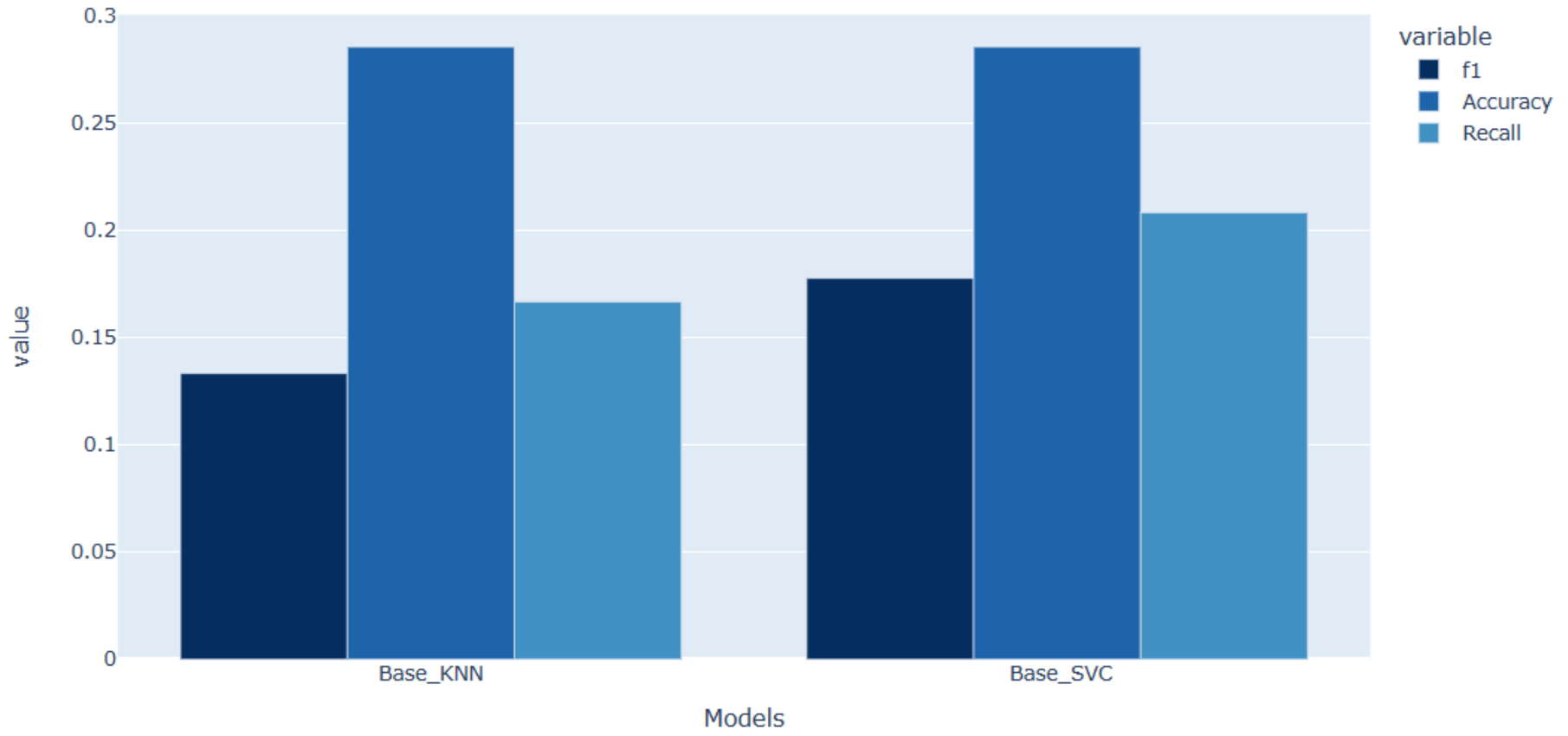
```
models = ['Base_KNN', ' Base_SVC']

f1 = [f1_score(y_val, yknn_pred, average= 'macro', zero_division= 0), f1_score(y_val, ysvc_pred, average= 'macro', zero
accuracy = [accuracy_score(y_val, yknn_pred), accuracy_score(y_val, ysvc_pred)]
recall = [recall_score(y_val, yknn_pred, average= 'macro'), recall_score(y_val, ysvc_pred, average= 'macro')]

metrics_df = pd.DataFrame({'Models': models, 'f1': f1, 'Accuracy': accuracy, 'Recall': recall})
metrics_df
```

3]:

	Models	f1	Accuracy	Recall
0	Base_KNN	0.133333	0.285714	0.166667
1	Base_SVC	0.177778	0.285714	0.208333

In [95]:

```
fig = px.bar(metrics_df, x='Models', y= ['f1', 'Accuracy', 'Recall'], barmode= 'group', color_discrete_sequence= px.colo
fig.show( )
```



After training our K-Nearest Neighbors (K-NN) and Support Vector Machines (SVM) models, we observe that SVM significantly outperforms K-NN in terms of the macro-averaged F1 score.

Given this performance difference, it makes sense to focus our efforts on the SVM model. We will proceed with fine-tuning this model to see if we can further improve the F1 score

MODEL TESTING

```
y_pred_test = best_clf.predict(X_test)
print('Test score with best model: ', f1_score(y_test, y_pred_test, average= 'macro', zero_division= 0))
```

```
Test score with best model:  0.7611111111111111
```

- Utilized the final model to predict test data.
- Achieved an F1 score of 0.7611.
- Significance:
  - Demonstrates the effectiveness of the model in real-world scenarios.
  - Indicates strong predictive accuracy and reliability.

CONCLUSION

- In conclusion, this presentation demonstrates the successful implementation of the anticipated model, complemented by a robust codebase that achieves high accuracy in disease prediction. This success finds the potential of data-driven approaches in revolutionizing healthcare and the importance of continued research and innovation in this field.

THANK YOU