Using SVM to classify stroke patients

Introduction to Machine Learning: Supervised Models

Data & Relevance

- 5,110 patients
- Attributes: gender, age, hypertension, heart_disease, ever_married, work_type, residence_type, avg_glucose_level, bmi, smoking_status, stroke
- Strokes account for 11% of global deaths (WHO)
- Help stroke diagnostics based on patient information

What problem is this model trying to solve?

- Create a model that can accurately classify patients as someone who has or has not had a stroke, then predict whether or not a patient has had a stroke based on the included attributes
- Early diagnostics for potential stroke candidates

Preprocessing

- BMI is the only column that has missing values, these will be filled using KNN
- Removal of id, work_type, and smoking_status column due to choosing to examine physical health attributes
- Changing relevant variables into binary variables (such as gender)
- Examining number of unique values in each column, many attributes are binary in this case

KNN

- Using K = 6, performing KNN using the KNNImpute from sklearn package
- Based on MSE analysis, 6 was the best number of neighbors for this dataset
  - Below is a code snippet of the MSE analysis and the MSEs are listed from K = 2 to K = 6

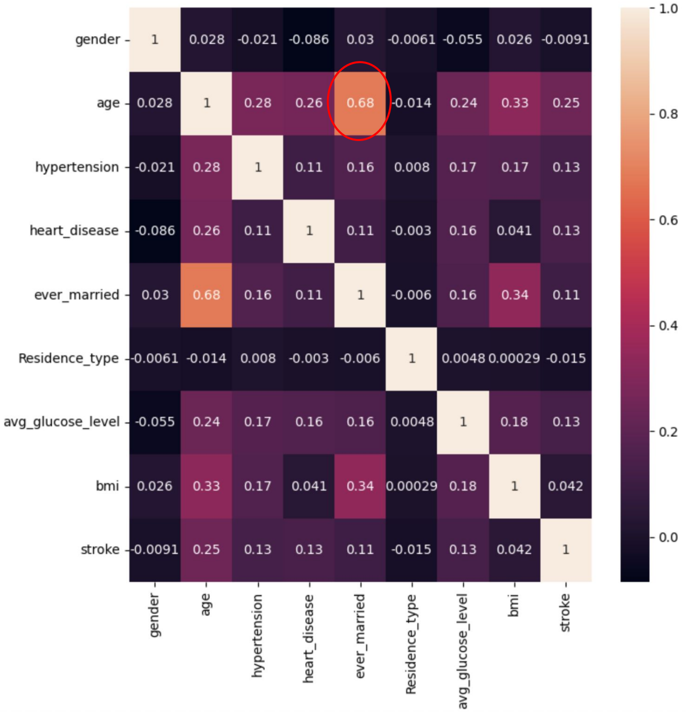
```
# Create validation set
indices = df.index.to_list()
val_indices = np.random.choice(indices, size=500, replace=False)
df_val = df.copy()
true_vals = df_val.loc[val_indices, 'bmi']
df_val.loc[val_indices, 'bmi'] = np.nan

# k = 2
knn2 = KNNImputer(n_neighbors=2)
df2 = pd.DataFrame(knn2.fit_transform(df_val), index=df_val.index, columns=df.columns)
imp2 = df2.loc[val_indices, 'bmi']
mask2 = ~imp2.isna() & ~true_vals.isna()
mse_2 = mean_squared_error(true_vals[mask2], imp2[mask2])
print(mse_2)
```

MSEs:

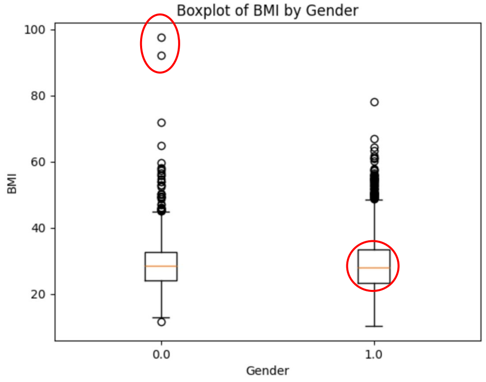
```
60.734065138888894
53.39005339506173
49.21945697916667
46.640755222222225
46.3706937345679
Best k based on MSE: 6
```

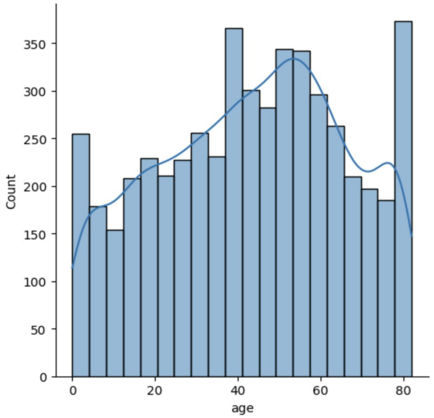
EDA

- ❑ Data is imbalanced, which may lead to biased outcomes
  - ❑ ~5% of patients have had a stroke
  - ❑ ~95% of patients have not had a reported stroke
- ❑ Most variables have weak correlations or no correlations
- ❑ Found that age and ever_married have the strongest correlation, so I will be removing ever_married

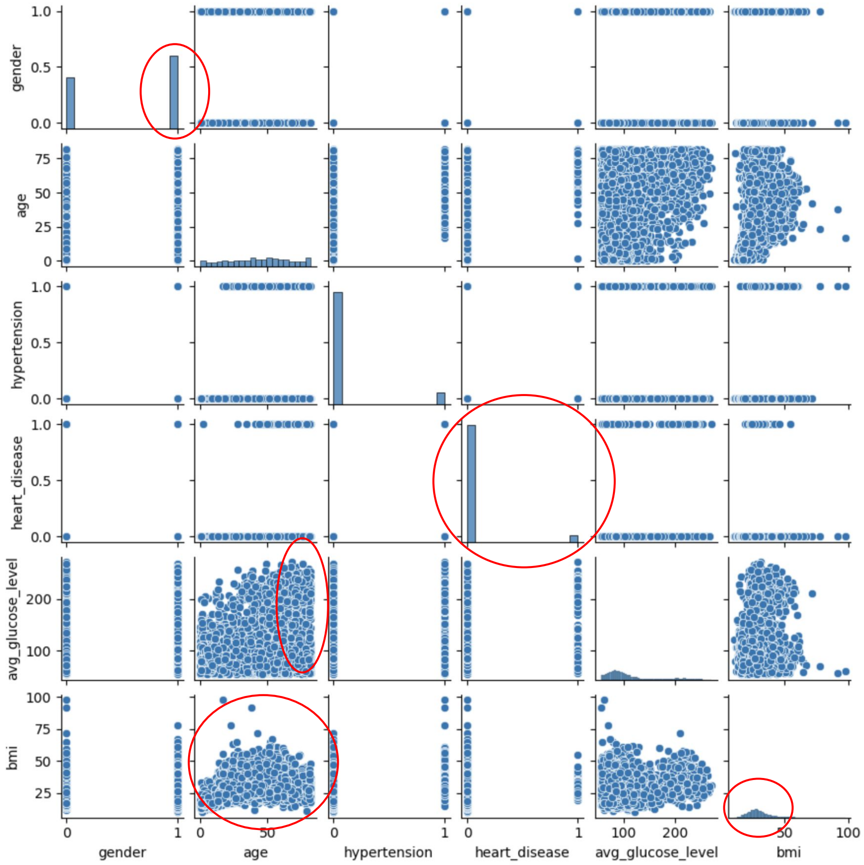


EDA Visualizations





- There are more female patients in the data set than men
- Men's BMI has more spread, but Women larger range and IQR than Men
- Patients are not only adults, but also include children
- Average Glucose is higher with patients that are older
- Age and BMI have a somewhat linear relationship
- Most patients do not have heart disease



Split and scale the data

- ❑ Scaled the data using python package: StandardScalar from sklearn

```
# Split into training and testing data
X = df.drop('stroke', axis=1)
y = df['stroke']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)
```

```
# Scale and standardize
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

SVM (First iteration)

- ❑ Run SVM model on the imbalanced data
  - ❑ Since it is imbalanced, we want to examine the F1 score to conclude how well the model handles false positives/negatives, and if it is considered a good fit for the data
- ❑ Using the 'class_weights' parameter of SVC to account for the imbalance
- ❑ Polynomial kernel worked the best with this iteration of the model
- ❑ Results:

```python
# SVM Train
svm = SVC(class_weight = 'balanced', kernel='poly', C=1)
cv_scores = cross_val_score(svm, X_train_scaled, y_train, cv=5, scoring='f1')
svm.fit(X_train_scaled, y_train)
y_pred = svm.predict(X_test_scaled)
accuracy = accuracy_score(y_test, y_pred)
print(classification_report(y_test, y_pred))
print(accuracy)
```

```
              precision    recall  f1-score   support

         0.0       0.98      0.77      0.86      1444
         1.0       0.16      0.69      0.25        89

    accuracy                           0.77      1533
   macro avg       0.57      0.73      0.56      1533
weighted avg       0.93      0.77      0.83      1533

0.7651663405088063
```

SVM (Second iteration)

- Attempted improvement: resampling the smaller class using SMOTE
- Oversampling the minority class may help us improve model function as it is only applied to the training data set
- Results:

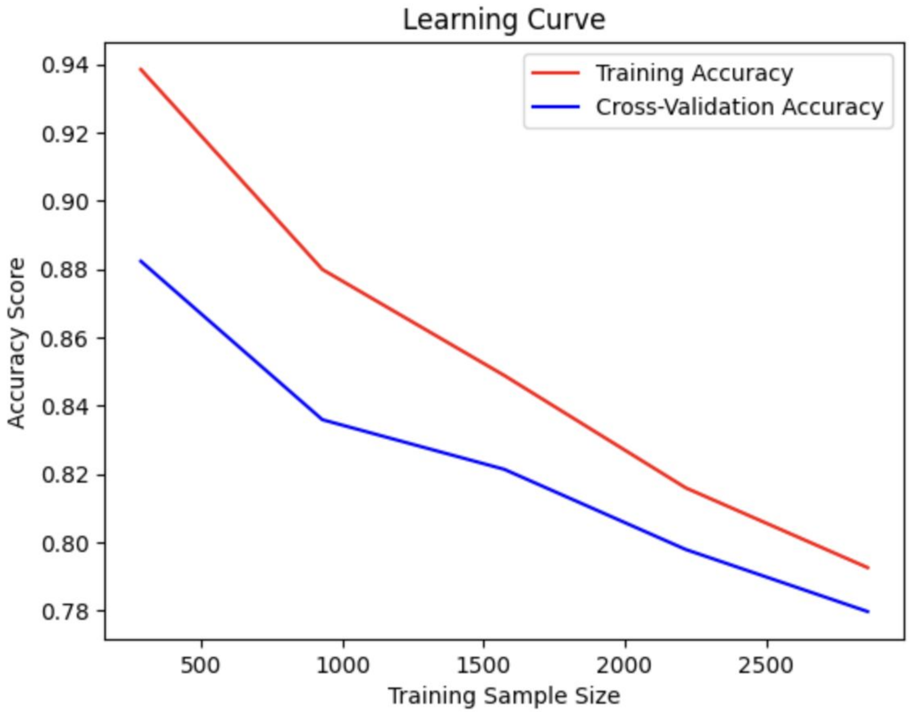
```python
# SVM with Resampling
from imblearn.over_sampling import SMOTE
smote = SMOTE(random_state=42)
X_resampled, y_resampled = smote.fit_resample(X_train_scaled, y_train)

# Try model with resampling
svm_r = SVC(class_weight = 'balanced', kernel='poly', C=1)
cv_scores = cross_val_score(svm_r, X_resampled, y_resampled, cv=5, scoring='f1')
svm_r.fit(X_resampled, y_resampled)
y_pred = svm_r.predict(X_test_scaled)
accuracy = accuracy_score(y_test, y_pred)
print(classification_report(y_test, y_pred))
print(accuracy)
```

	precision	recall	f1-score	support
0.0	0.98	0.78	0.87	1444
1.0	0.16	0.67	0.26	89
accuracy			0.78	1533
macro avg	0.57	0.73	0.56	1533
weighted avg	0.93	0.78	0.83	1533

0.7775603392041748

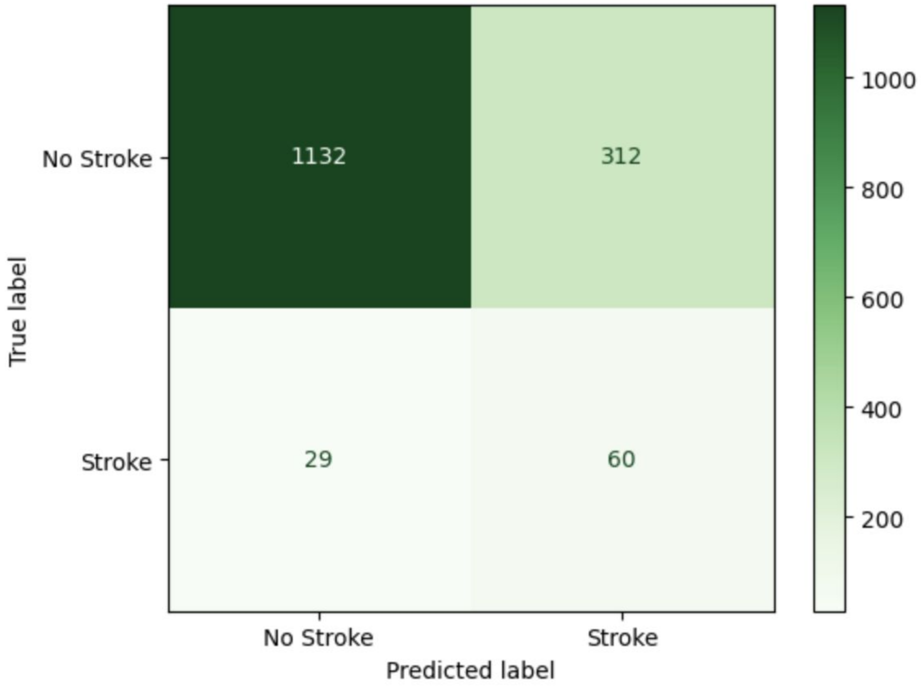
Learning Curve & Conclusions

- ❑ Based on the learning curve, the model may benefit from a smaller sample size
- ❑ This particular model is overfitting and capturing unnecessary noise



Confusion Matrix & Considerations

- The model is better at fitting patients that have not had a stroke (97.5% accurate)
- Minority class is very poorly predicted (16%)
- Data imbalance may make SVM a poor fit for this data set, especially when looking at physical attributes of patients and not outside parameters (married, job, residence)



Conclusion

- This model did not accurately solve the problem of predicting future stroke patients
- Random forest regression may have been a better fit for this data
- Other model applications should be explored for this data set
- More data on the patients such as whether or not they have diabetes, weight, or average number of hours worked per week may help improve the model

Notebook & Github Link

- ❑ Submitted Notebook can be found on Github and submitted with this presentation
- ❑ Github link