Project Overview - Breast Tumor Prediction using ML

1. What are we trying to find out?

The main question we want to answer through this project is: **"Can we accurately predict whether a breast cancer diagnosis is malignant or benign, based on specific characteristics of cell nuclei measured in the dataset?"**

2. What do we already know?

- The dataset contains various measurements of cell nuclei features extracted from breast cancer images.
- Each sample in the dataset is labeled as either 'M' (Malignant) or 'B' (Benign).
- Features include metrics like radius, texture, perimeter, area, smoothness, compactness, concavity, and symmetry, among others.
- Previous studies have demonstrated that these features can be predictive of cancer types when properly analyzed.

  The reference for the statement above *can be found in several studies* that have explored the morphological features of breast cancer cells and their significance in diagnosis. One of the most cited and foundational studies is: **Street, W. N., Wolberg, W. H., & Mangasarian, O. L. (1993). "Nuclear feature extraction for breast tumor diagnosis."** SPIE Vol. 1905, Biomedical Image Processing and Biomedical Visualization, 861-870.

3. What are we aiming to achieve?

Our goal is to create a highly accurate ML model for classifying diagnoses, aiming for high sensitivity and specificity. This model will be valuable for enhancing medical diagnostic accuracy and decision-making.

4. What factors affect our results?

Key factors influencing the model's performance include:

- **Data quality**: Inaccurate or incomplete data could reduce model effectiveness. Noises or errors in the measurements could affect model accuracy.
- **Feature importance**: Some features may contribute more to the model’s ability to distinguish between cancer types.
- **Model choice and complexity**: Selecting the right model and tuning its complexity is crucial to avoid overfitting or underfitting.

5. Is there something new we can use? – new ideas or methods

Data Augmentation and Synthesis (Enrich small dataset):

- **Synthetic Data Generation**: Tools like GANs (Generative Adversarial Networks) can create synthetic samples, which can be particularly useful when our dataset is small.
- **Augmentation Techniques**: Techniques like flipping, rotating, or adding noise can increase the diversity of the training dataset, helping the model generalize better.

1. Data Preparation

Here is a brief overview of the dataset:

- Columns and Rows:

  df.shape() => (569, 33)

- Target Variable: The ‘diagnosis’ column indicates the type of the tumer ('M’ or ‘B’). This column is of object type and is converted to int64

  df['diagnosis'] = df['diagnosis'].replace({'M': 1, 'B': 0}).

- Empty Column Drop: A column named ‘Unnamed: 32’ has zero values

  df = df.drop('Unnamed: 32', axis=1)

2. EDA – Explanatory Data Analytics

- Potential Issues:

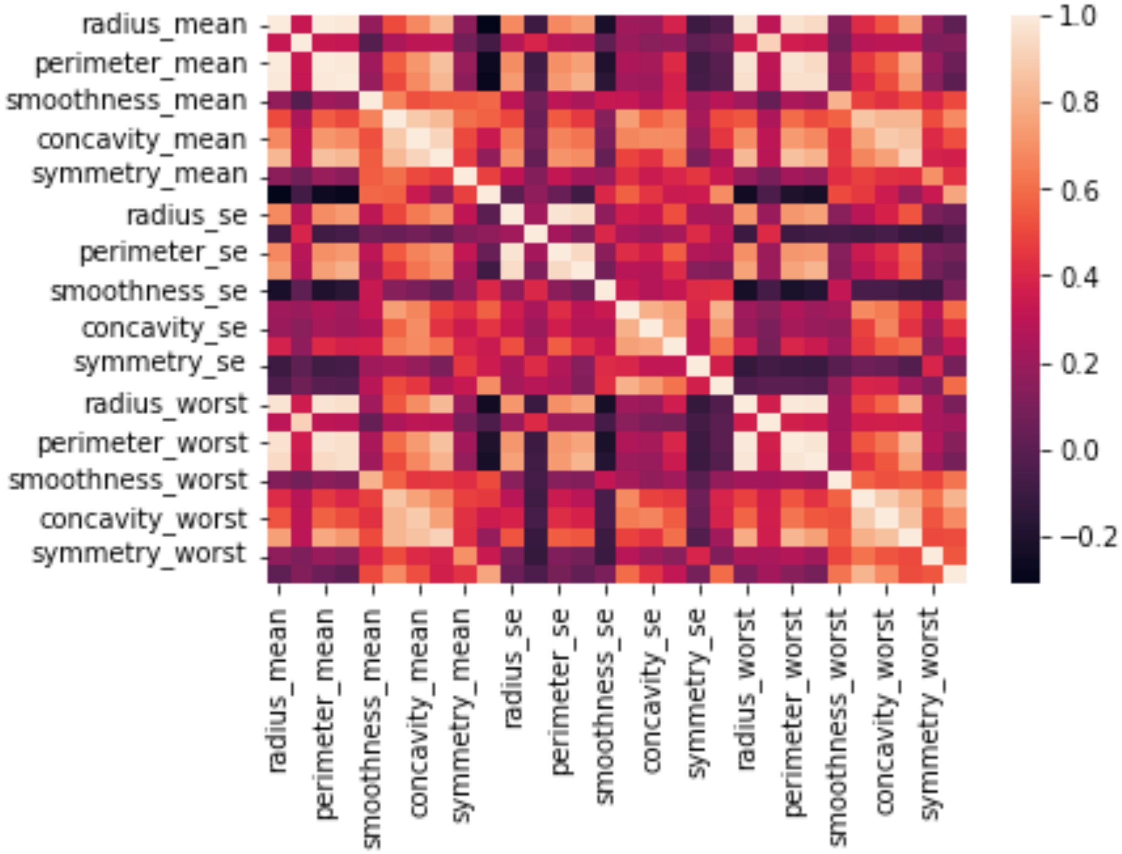
  i. Several features have high skewness, indicating that they are not normally distributed. Highly skewed data can impact models that assume normality, such as linear regression or logistic regression, leading to poor performance.

  ii. The ‘id’ column consists of unique values that serve as identifiers for each observation in the dataset. Including this feature in model training could potentially lead to overfitting.

- Correlation:

  Based on the correlation matrix, there are several pairs of features with strong multicollinearity. Here are some of the most notable pairs:

- perimeter_mean and radius_mean: Correlation = 0.9979
- radius_worst and perimeter_worst: Correlation = 0.9937
- radius_mean and area_mean: Correlation = 0.9874
- perimeter_mean and area_mean: Correlation = 0.9865
- radius_worst and area_worst: Correlation = 0.9818
- perimeter_worst and area_worst: Correlation = 0.9865
- concave points_mean and concavity_mean: Correlation = 0.9218
- concave points_worst and concavity_worst: Correlation = 0.9168



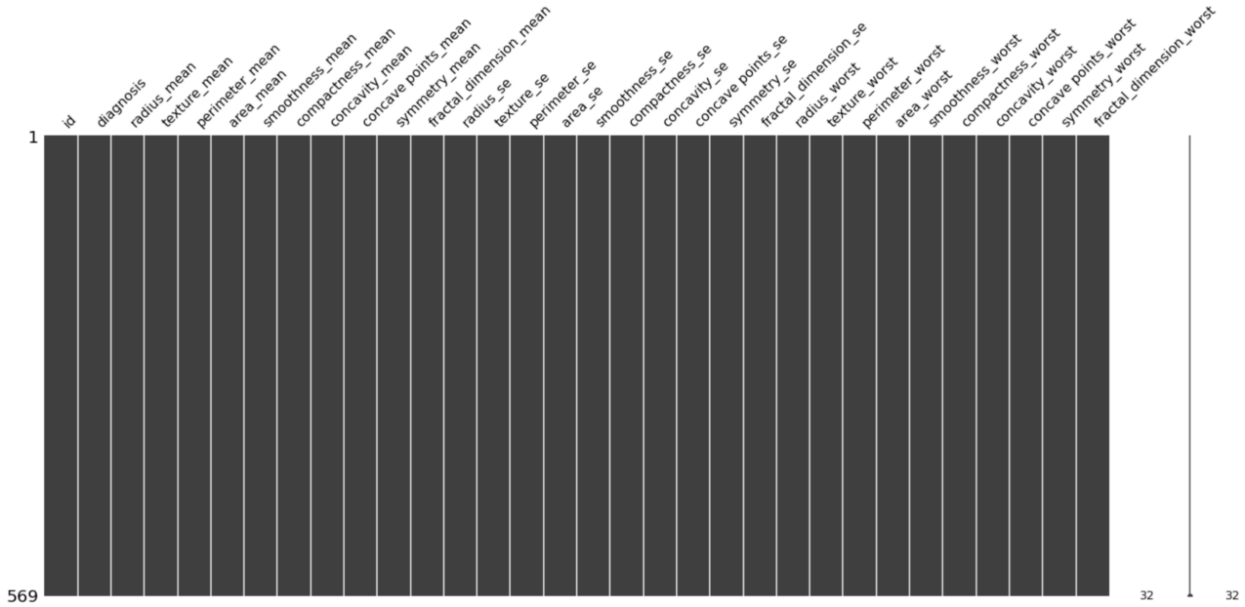
3. Data Cleansing

- Outliers:

  Based on IQR identifing of outliers, no outliers were detected for this dataset.

- Missing Values:

  Based on checking distribution and correlation changes and by running Missingno diagram, there're no missing values.

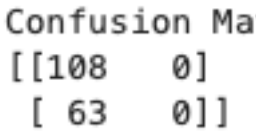


4. One-Hot Encoding

For current dataset, since the only categorical feature 'diagnosis' has already been encoded as a binary variable, **one-hot encoding is not needed.** (Imputation with KNN - fill in missing values – is not needed)

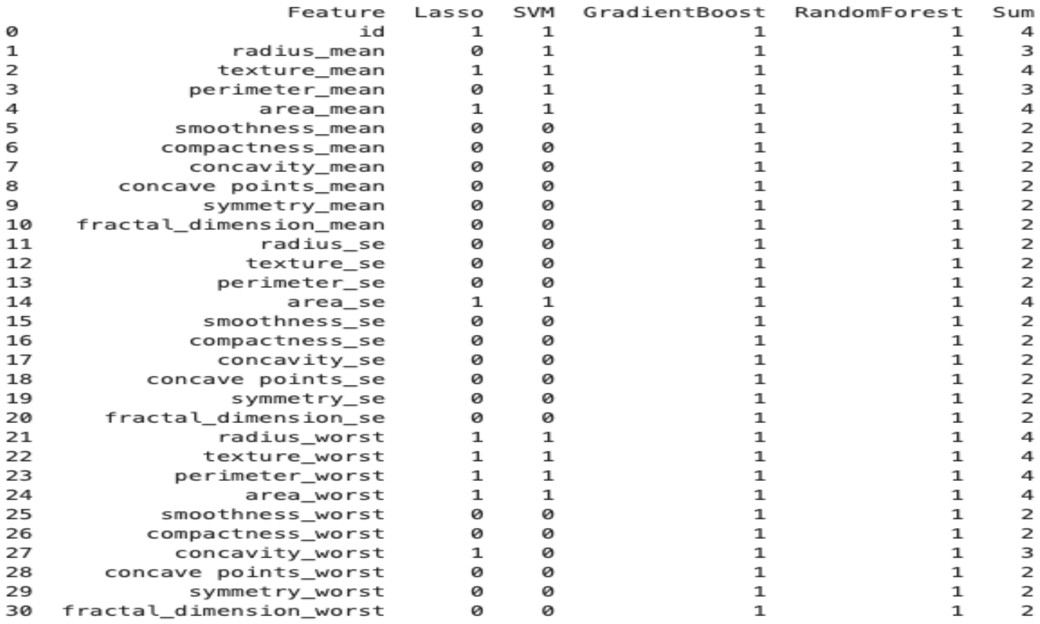
5. Features Selection

- Remove Irrelevant Features: The ‘id’ feature was removed as it does not contribute to predicting the target variable and, even worse, it can cause overfitting!

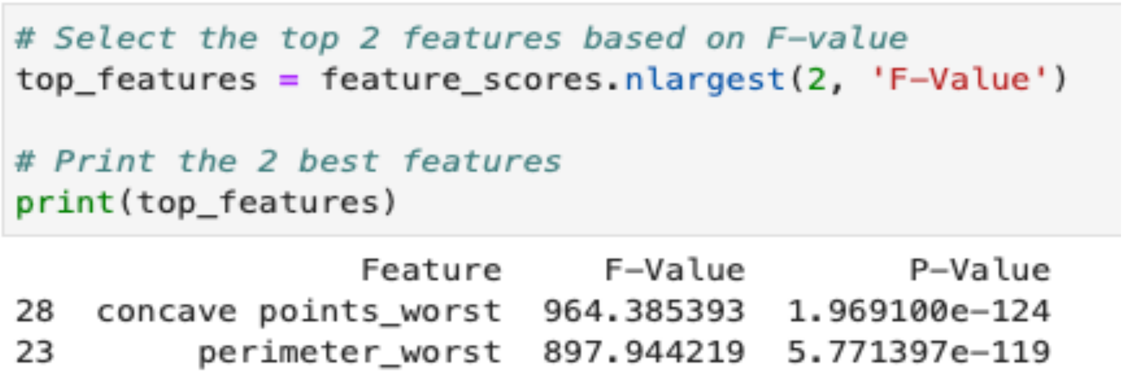


```
Confusion Matrix:
[[108   0]
 [ 63   0]]
```

- Remove Redundant Features: Based on the correlation matrix, some features like radius_mean, perimeter_mean, and area_mean are highly correlated.
- Multivariable Analysis: Based on DataFrame creation with most valuable variables, we might need to leave only features importance by 3 or more models, but it is *recommaneded to have at least 30 features*, so we won’t remove features in this case.



```
                   Feature  Lasso  SVM  GradientBoost  RandomForest  Sum
0                       id      1    1              1             1    4
1              radius_mean      0    1              1             1    3
2             texture_mean      1    1              1             1    4
3           perimeter_mean      0    1              1             1    3
4                area_mean      1    1              1             1    4
5          smoothness_mean      0    0              1             1    2
6         compactness_mean      0    0              1             1    2
7           concavity_mean      0    0              1             1    2
8      concave points_mean      0    0              1             1    2
9            symmetry_mean      0    0              1             1    2
10  fractal_dimension_mean      0    0              1             1    2
11               radius_se      0    0              1             1    2
12              texture_se      0    0              1             1    2
13            perimeter_se      0    0              1             1    2
14                 area_se      1    1              1             1    4
15           smoothness_se      0    0              1             1    2
16          compactness_se      0    0              1             1    2
17            concavity_se      0    0              1             1    2
18       concave points_se      0    0              1             1    2
19             symmetry_se      0    0              1             1    2
20    fractal_dimension_se      0    0              1             1    2
21            radius_worst      1    1              1             1    4
22           texture_worst      1    1              1             1    4
23         perimeter_worst      1    1              1             1    4
24              area_worst      1    1              1             1    4
25        smoothness_worst      0    0              1             1    2
26       compactness_worst      0    0              1             1    2
27         concavity_worst      1    0              1             1    3
28    concave points_worst      0    0              1             1    2
29          symmetry_worst      0    0              1             1    2
30 fractal_dimension_worst      0    0              1             1    2
```



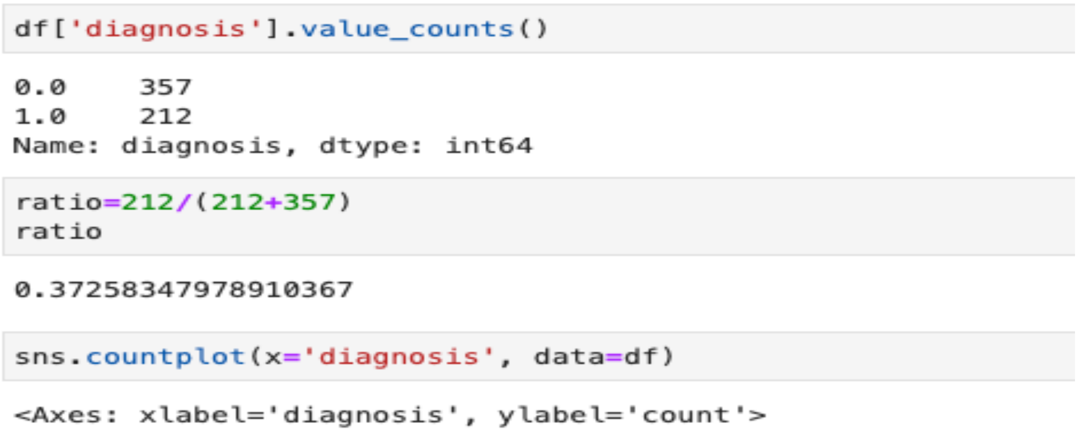
```
# Select the top 2 features based on F-value
top_features = feature_scores.nlargest(2, 'F-Value')

# Print the 2 best features
print(top_features)
```

```
                 Feature     F-Value        P-Value
28  concave points_worst  964.385393  1.969100e-124
23       perimeter_worst  897.944219  5.771397e-119
```

6. **Imbalanced Data**

   For current dataset, since the target feature 'diagnosis' is split as 63%:37% by value counts, no Imbalance is needed. Imbalanced data is need on 80%:20%. (In case we would need it, One of ROS, RUS, SMOTE or SMOTETomek techniques with the most optimal results will be used)



```
df['diagnosis'].value_counts()
```

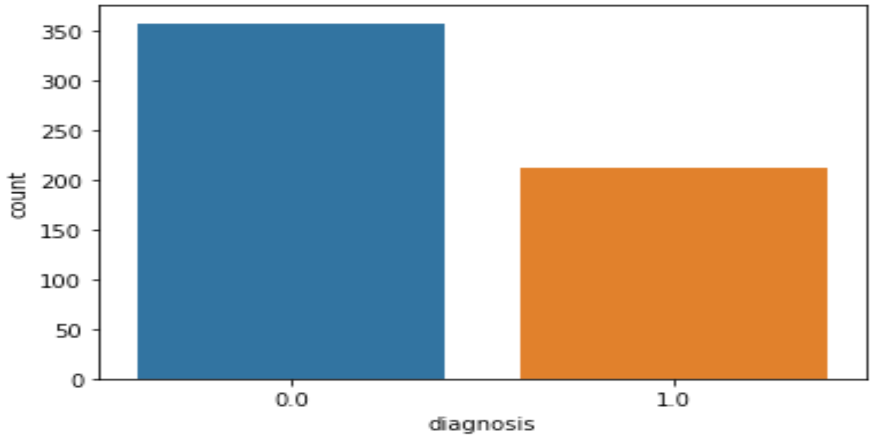
```
0.0    357
1.0    212
Name: diagnosis, dtype: int64
```

```
ratio=212/(212+357)
ratio
```

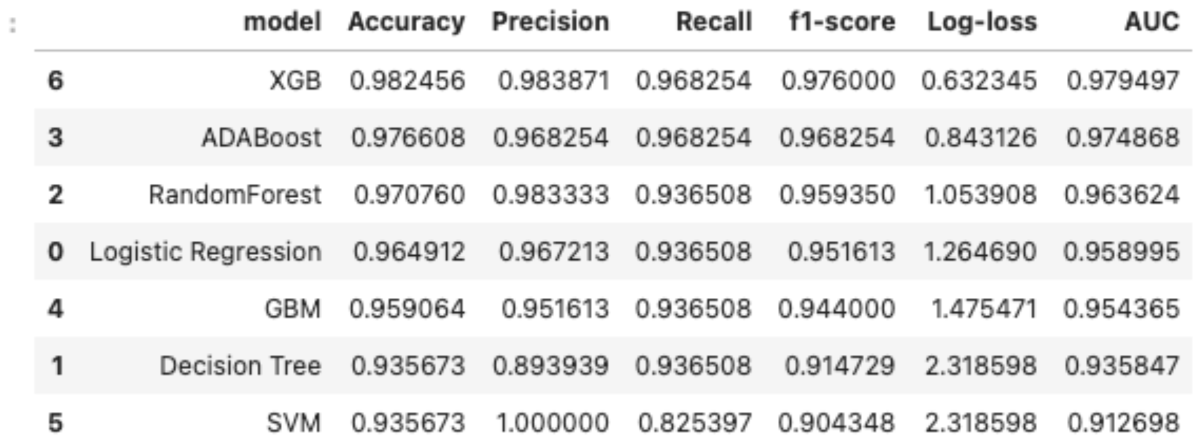
```
0.37258347978910367
```

```
sns.countplot(x='diagnosis', data=df)
```

```
<Axes: xlabel='diagnosis', ylabel='count'>
```



7. **Model Selection**
   Selecting the right model for the dataset is crucial for achieving good predictive performance. Given the dataset involves predicting a binary outcome (malignant vs. benign tumors) based on numerical features, here's a list of relevant models and the performance metrics:



	model	Accuracy	Precision	Recall	f1-score	Log-loss	AUC
6	XGB	0.982456	0.983871	0.968254	0.976000	0.632345	0.979497
3	ADABoost	0.976608	0.968254	0.968254	0.968254	0.843126	0.974868
2	RandomForest	0.970760	0.983333	0.936508	0.959350	1.053908	0.963624
0	Logistic Regression	0.964912	0.967213	0.936508	0.951613	1.264690	0.958995
4	GBM	0.959064	0.951613	0.936508	0.944000	1.475471	0.954365
1	Decision Tree	0.935673	0.893939	0.936508	0.914729	2.318598	0.935847
5	SVM	0.935673	1.000000	0.825397	0.904348	2.318598	0.912698

Conclution

XGBoost achieved the highest accuracy (98.25%), indicating that it correctly classified 98.25% of all cases

Summary

Multiple models were evaluated in order to predict whether a tumor is malignant or benign using various metrics. Below is an explanation of each metric used for the results above.

- **Accuracy**: The proportion of correctly classified instances out of all instances. It provides an overall measure of how well the model performs.

- **Precision**: The proportion of true positive predictions among all positive predictions. High precision indicates that when the model predicts a tumor as malignant, it is likely to be correct.

- **Recall (Sensitivity)**: The proportion of true positive predictions among all actual positive cases. High recall means the model is good at identifying malignant tumors and does not miss many cases.

- **F1-Score**: The harmonic mean of precision and recall. It provides a balance between precision and recall.

- **Log-loss**: A metric that measures the uncertainty of the model’s predictions by comparing the predicted probability of the true class to 1. Lower log-loss indicates better-calibrated probabilities and higher confidence in predictions.

- **AUC (Area Under the ROC Curve)**: A metric that measures the model's ability to distinguish between classes across all possible threshold levels. A higher AUC indicates better performance in distinguishing between malignant and benign tumors.

ML System Implementation and Integration

1. **Data Pipeline Setup**

   The pipeline should handle data collection, preprocessing, and storage:

   - Set up processes to automatically collect new data from existing databases or other sources.
   - Ensure that the data is cleaned, normalized, and transformed as required by the ML model.
   - Store the preprocessed data in a structured format that the ML model can access.

2. **Model Deployment**

   Deploying the machine learning model for making it available for use in the production environment:

   - Using a model serving framework (e.g., TensorFlow Serving, MLflow, FastAPI) to expose the model as a web service or API endpoint.
   - Package the model and its dependencies using containerization technologies like Docker. This ensures that the model runs consistently across different environments.

3. **Integration with Existing Systems**

   Once the model is deployed, it needs to be integrated into the existing systems:

- Modify existing applications to send data to the ML model via APIs and receive predictions. For instance, a hospital's electronic health record (EHR) system might be integrated with the ML model to automatically analyze patient data and provide risk scores.
- Update the user interfaces of existing systems to display model predictions and insights.
- Set up automation to trigger model predictions based on specific events or time intervals. For example, a batch process could be scheduled to run predictions on new patient data every night.

Users and Benefits from the ML System

The machine learning system will be used by a variety of stakeholders, including oncologists, radiologists, pathologists, healthcare administrators, researchers and patients.

It will improve diagnostic accuracy and efficiency, enhance decision-making, reduce costs, and ultimately lead to better patient outcomes and satisfaction.

By integrating this technology, all these stakeholders stand to benefit from improved healthcare delivery and patient care.

The primary users of the ML system are those who will interact directly with the model outputs to perform their tasks more efficiently and accurately. The primary users might include:

- **Oncologists and Radiologists**:
  - **How They Will Use the System**: Oncologists and radiologists will use the ML model to assist in diagnosing cancerous tumors. The model can analyze imaging data or patient records to predict whether a tumor is benign or malignant.
  - **Benefits**: The system helps in making faster, more accurate diagnoses by providing a second opinion based on large datasets. It can reduce the likelihood of misdiagnosis and ensure that critical cases are flagged for further review, improving patient outcomes.
- **Pathologists**:
  - **How They Will Use the System**: Pathologists can use the ML model to analyze biopsy samples or other clinical data, helping them to identify cancerous cells more quickly and accurately.
  - **Benefits**: The system can streamline the pathology workflow by highlighting areas of interest or concern, reducing the time spent on each case and allowing pathologists to focus on more complex analyses.