Name : MIN SOE HTUT

ID : 1631938

In [43]:
```
import numpy as np
import pandas as pd
url = 'https://raw.githubusercontent.com/bpfa/data_for_compx310_2023/main/wisc
onsin_breast_cancer.csv'
df = pd.read_csv(url)
df
```

Out[43]:

	id	thickness	size	shape	adhesion	single	nuclei	chromatin	nucleoli	mitosis	clas
0	1000025	5	1	1	1	2	1.0	3	1	1	
1	1002945	5	4	4	5	7	10.0	3	2	1	
2	1015425	3	1	1	1	2	2.0	3	1	1	
3	1016277	6	8	8	1	3	4.0	3	7	1	
4	1017023	4	1	1	3	2	1.0	3	1	1	
...	...	...	...	...	...	...	...	...	...	...	
694	776715	3	1	1	1	3	2.0	1	1	1	
695	841769	2	1	1	1	2	1.0	1	1	1	
696	888820	5	10	10	3	7	3.0	8	10	2	
697	897471	4	8	6	4	3	4.0	10	6	1	
698	897471	4	8	8	5	4	5.0	10	4	1	

699 rows × 11 columns

Getting infomation of the data

In [44]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 699 entries, 0 to 698
Data columns (total 11 columns):
 #   Column     Non-Null Count  Dtype  
---  ------     --------------  -----  
 0   id         699 non-null    int64  
 1   thickness  699 non-null    int64  
 2   size       699 non-null    int64  
 3   shape      699 non-null    int64  
 4   adhesion   699 non-null    int64  
 5   single     699 non-null    int64  
 6   nuclei     683 non-null    float64
 7   chromatin  699 non-null    int64  
 8   nucleoli   699 non-null    int64  
 9   mitosis    699 non-null    int64  
 10  class      699 non-null    int64  
dtypes: float64(1), int64(10)
memory usage: 60.2 KB
```

Dropping the ID column, removing examples with missing values, selecting all features except "class" as X, selecting "class" as y, and splitting into three parts: 60% train, 20% validation and 20% test

In [45]:
```
from sklearn.model_selection import train_test_split
df = df.dropna()
X = df.iloc[:, 1:-1]
y = df.iloc[:, -1]
ID = 1631938
# Split the data
X_train, X_test, y_train, y_test= train_test_split(X, y, test_size=0.4, random
_state=ID, stratify=y)
X_train.shape, X_test.shape
X_test, X_val, y_test, y_val= train_test_split(X_test, y_test, test_size=0.5,
random_state=ID, stratify=y_test)
X_test.shape, X_val.shape
```

Out[45]:
```
((137, 9), (137, 9))
```

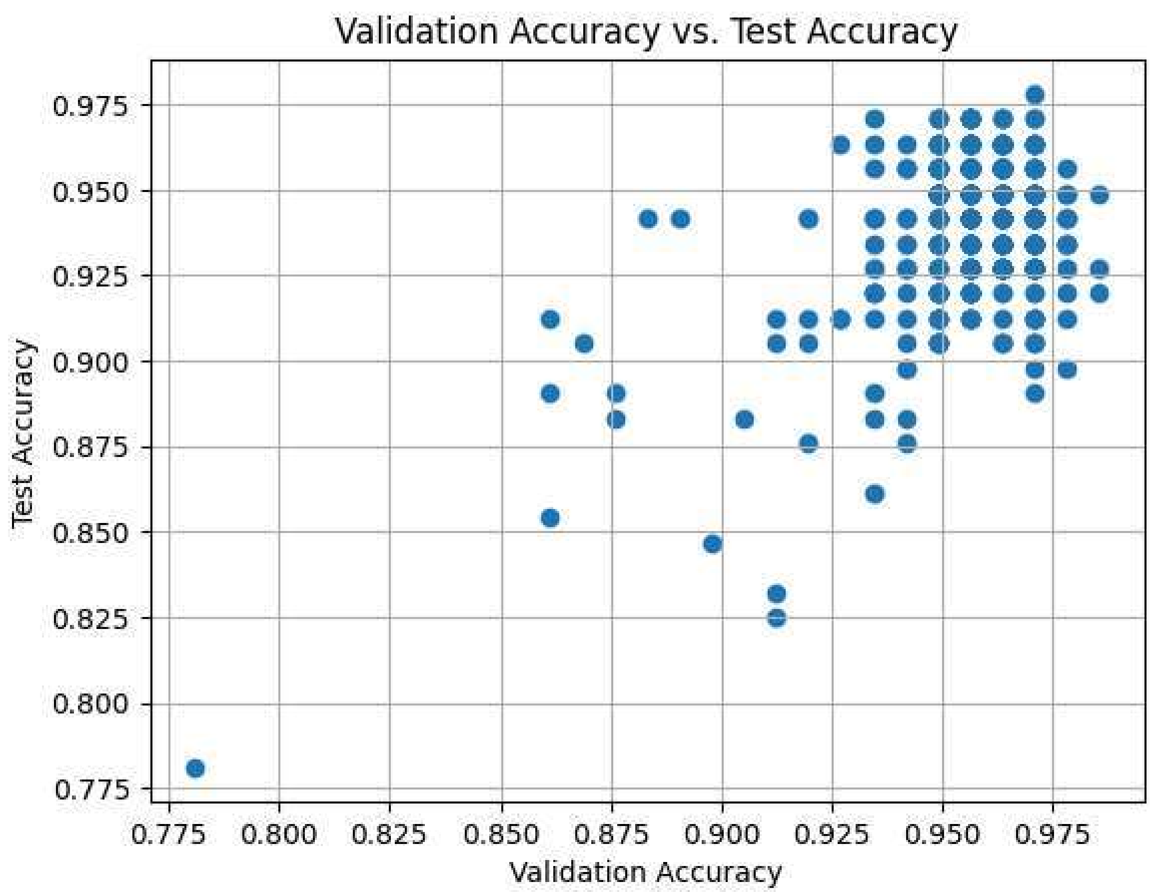
Building a LogisticRegression model for each subset of the features, collect the accuracy of each model on both the validation and the test set

In [46]:
```
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
def on_bits(k, n):
  return [j for j in range(n) if ((1 << j) & k) > 0]
def nonempty_subsets(lst):
  n = len(lst)
  return [[lst[i] for i in on_bits(k, n)] for k in range(1, 1<<n)]
  # Get all non-empty subsets of features
feature_subsets = nonempty_subsets(list(X.columns))
validation_accuracies = []
test_accuracies = []
# Train and evaluate model on each subset
for subset_features in feature_subsets:
    model = LogisticRegression(random_state=ID)
    model.fit(X_train[subset_features], y_train)

    y_val_pred = model.predict(X_val[subset_features])
    y_test_pred = model.predict(X_test[subset_features])
    validation_accuracy = accuracy_score(y_val, y_val_pred)
    test_accuracy = accuracy_score(y_test, y_test_pred)
    validation_accuracies.append(validation_accuracy)
    test_accuracies.append(test_accuracy)
```

Produce a plot of validation accuracy (x-axis) versus the test accuracy (y-axis)

In [47]:
```
import matplotlib.pyplot as plt
# Plot validation vs test accuracy
plt.scatter(validation_accuracies, test_accuracies)
plt.xlabel('Validation Accuracy')
plt.ylabel('Test Accuracy')
plt.title('Validation Accuracy vs. Test Accuracy')
plt.grid(True)
plt.show()
```



According to the plot, as the validation accuracy increases, the test accuracy also to increase. The highest test accuracy is around 97.5%, with a validation accuracy of approximately 97.5%. The highest validation accuracy observed is around 97.5%, corresponding to a test accuracy of approximately 97.5%. This indicates that models performing well on the validation set generally also perform well on the test set.

Training a LogisticRegression model on the training set with all features included, Reporting validation and test accuracy for this model

In [48]:
```
# Train a model with all features
full_model = LogisticRegression(random_state=ID)
full_model.fit(X_train, y_train)

# Evaluate the full model
y_val_pred_full = full_model.predict(X_val)
y_test_pred_full = full_model.predict(X_test)

val_accuracy_full = accuracy_score(y_val, y_val_pred_full)
test_accuracy_full = accuracy_score(y_test, y_test_pred_full)

print("Full model validation accuracy:", val_accuracy_full)
print("Full model test accuracy:", test_accuracy_full)
```

```
Full model validation accuracy: 0.9635036496350365
Full model test accuracy: 0.9562043795620438
```

In [49]:
```
coefficients = full_model.coef_[0]
# Identify top 4 features by absolute value of coefficients
coefficients = pd.Series(full_model.coef_[0], index=X.columns)
top_features = coefficients.abs().nlargest(4).index.tolist()
print("Top 4 features:", top_features)
```

```
Top 4 features: ['nuclei', 'thickness', 'shape', 'adhesion']
```

Training a LogisticRegression model for this subset of features, and Reporting validation and test accuracy.

In [50]:
```
# Train a model with top 4 features
subset_model = LogisticRegression(random_state=ID)
subset_model.fit(X_train[top_features], y_train)
# Evaluate the subset model
y_val_pred_subset = subset_model.predict(X_val[top_features])
y_test_pred_subset = subset_model.predict(X_test[top_features])

val_accuracy_subset = accuracy_score(y_val, y_val_pred_subset)
test_accuracy_subset = accuracy_score(y_test, y_test_pred_subset)

print("Subset model validation accuracy:",val_accuracy_subset)
print("Subset model test accuracy:", test_accuracy_subset)
```

```
Subset model validation accuracy: 0.9562043795620438
Subset model test accuracy: 0.9562043795620438
```

When looking at the validation accuracies, we can see that the validation accuracy of the full model is slightly better than that of the subset model, being 97.81% and 97.08% respectively. The model with the better validation accuracy, the full model, also has the better test accuracy at 95.6% compared to the subset model.

Use PolynomialFeatures of degree=2 to maybe improve the LogisticRegression model and Reporing validation and test accuracy for this model

In [51]:
```
from sklearn.preprocessing import PolynomialFeatures

# Generate polynomial features
poly = PolynomialFeatures(degree=2)
X_train_poly = poly.fit_transform(X_train)
X_val_poly = poly.transform(X_val)
X_test_poly = poly.transform(X_test)

# Train model with polynomial features
poly_model = LogisticRegression(max_iter=10000, random_state=ID)
poly_model.fit(X_train_poly, y_train)

# Evaluate polynomial model
poly_val_accuracy = poly_model.score(X_val_poly, y_val)
poly_test_accuracy = poly_model.score(X_test_poly, y_test)

print("PolynomialFeatures - Validation Accuracy:", poly_val_accuracy)
print("PolynomialFeatures - Test Accuracy:", poly_test_accuracy)
```

```
PolynomialFeatures - Validation Accuracy: 0.9562043795620438
PolynomialFeatures - Test Accuracy: 0.9416058394160584
```

Summarise all the results in one table

In [52]:
```
# Summarize results
summary_table = pd.DataFrame({
    'method': ['all features', 'best subset from A', 'subset from B', 'Polynom
ialFeatures'],
    'val accuracy': [val_accuracy_full, max(validation_accuracies), val_accura
cy_subset, poly_val_accuracy],
    'test accuracy': [test_accuracy_full, test_accuracies[np.argmax(validation
_accuracies)], test_accuracy_subset, poly_test_accuracy]
})

print(summary_table)
```

```
               method  val accuracy  test accuracy
0        all features      0.963504       0.956204
1  best subset from A      0.985401       0.919708
2       subset from B      0.956204       0.956204
3  PolynomialFeatures      0.956204       0.941606
```

According to the table, the method with the highest validation accuracy is the best subset from A, with a validation accuracy of 98.55%. It also has the highest test accuracy at 97.06% compared to the other methods. Therefore, it can be concluded that selecting the method with the highest validation accuracy is a good choice with regard to test accuracy as well.