



Do it for all of the rest

Let's go through **ALL remaining experiments** (2–7) with:

- Code-based + theory/manual-based viva questions
- *Answers provided*
- Sample code/query where relevant
- Each divided into Basic, Intermediate, and Advanced

Experiment 2: Regression Analysis for Uber Ride Price Prediction

BASIC

Q1: What is the main goal of regression analysis in this experiment?

A: To predict the price (fare) of an Uber ride from pickup and drop-off details using linear regression models (including Ridge/Lasso), after appropriate preprocessing.^[1]

Q2: What is an outlier, and why is handling them important here?

A: An outlier is a data point far removed from others. Outliers can distort regression coefficients and worsen prediction performance.

Q3: What are the primary metrics to evaluate your model?

A: \$ R^2 \$ (coefficient of determination) and RMSE (Root Mean Squared Error) are key evaluation metrics for regression.

INTERMEDIATE

Q4: How is the Haversine formula useful in this dataset?

A: It computes the straight-line ("as-the-crow-flies") distance between the pickup and dropoff points using their latitude and longitude, which is an important feature for predicting fare.

Code Query:

Calculate distance using the Haversine formula in Python, for two points (lat1, lon1) and (lat2, lon2).

```
from math import radians, cos, sin, asin, sqrt

def haversine(lat1, lon1, lat2, lon2):
    R = 6371 # Earth's radius in kilometers
    dlat = radians(lat2 - lat1)
```

```

dlon = radians(lon2 - lon1)
a = sin(dlat/2)**2 + cos(radians(lat1)) * cos(radians(lat2)) * sin(dlon/2)**2
c = 2 * asin(sqrt(a))
return R * c

```

Q5: What is the effect of multicollinearity, and how do Ridge and Lasso address it?

A: Multicollinearity occurs when features are highly correlated, which can make regression unstable. Ridge regression penalizes large coefficients, shrinking them but retaining all features; Lasso can shrink some coefficients to zero, effectively performing feature selection.

ADVANCED

Q6: You built Linear, Ridge, and Lasso regression models. If Ridge improves test accuracy but Lasso sets many coefficients to zero, how do you interpret this?

A: Ridge handles correlated features well by shrinking their coefficients, while Lasso eliminates irrelevant features (coefficients set to zero), simplifying the model—even if those features are correlated.

Q7: Your RMSE is higher than your fare range—what might have gone wrong?

A: Possible issues: serious outliers, a non-linear relationship not captured by linear models, incorrect features, or problems in preprocessing (missing values, etc.)

Q8: Given a row with missing fare, how could you estimate/impute a value programmatically?

```

# Using predicted value from the trained model
sample = X.loc[row_index_with_missing_fare].values.reshape(1, -1)
predicted_fare = model.predict(sample)

```

Experiment 3: SVM for Handwritten Digit Classification

BASIC

Q1: What is the central algorithm used, and what is its primary objective?

A: Support Vector Machine (SVM), used to classify images of handwritten digits (0–9) into their correct numeric label.^[1]

Q2: What makes SVMs suitable for this task?

A: They work well on high-dimensional data (e.g., images), handle non-linear boundaries via kernels, and perform well with small/medium datasets.

Q3: What is a hyperplane in the context of SVM?

A: It's the decision boundary that separates different classes with the maximum possible margin.

INTERMEDIATE

Q4: What is the role of kernels in SVM?

A: Kernels enable SVMs to classify data that is not linearly separable by transforming it to higher dimensions where a linear separator can be found.

Q5: What metrics do you use for SVM performance?

A: Accuracy, Precision, Recall, F1-score, and confusion matrix.

Code Query:

Show code to plot the confusion matrix for predictions on test data.

```
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay
import matplotlib.pyplot as plt

cm = confusion_matrix(y_test, y_pred)
disp = ConfusionMatrixDisplay(confusion_matrix=cm)
disp.plot()
plt.show()
```

ADVANCED

Q6: Explain how SVM handles multi-class classification tasks like this one.

A: SVM implements multi-class classification by combining several binary classifiers, using "one-vs-one" or "one-vs-rest" strategies.

Q7: What practical challenges might you face with SVM on the digit data?

A: Class overlap in feature space; choosing proper kernel and hyperparameters; computation is costly for large datasets.

Q8: How would you use cross-validation to select SVM hyperparameters in code?

```
from sklearn.model_selection import GridSearchCV
from sklearn.svm import SVC

params = {'C':[0.1,1,10], 'kernel':['linear','rbf']}
grid = GridSearchCV(SVC(), params, cv=5)
grid.fit(X_train, y_train)
print(grid.best_params_)
```

Experiment 4: K-Means Clustering on Iris Dataset

BASIC

Q1: What is clustering?

A: It's an unsupervised learning method that groups data points with similar features into clusters, with no label information.^[1]

Q2: How does the elbow method help in K-Means?

A: It plots within-cluster-sum-of-squares (WCSS) for different values of k; the "elbow" point indicates an optimal k.^[1]

Code Query:

Write code to display the elbow plot.

```
from sklearn.cluster import KMeans
import matplotlib.pyplot as plt

WCSS = []
for k in range(1, 11):
    kmeans = KMeans(n_clusters=k, random_state=42).fit(X)
    WCSS.append(kmeans.inertia_)

plt.plot(range(1, 11), WCSS, 'bo-')
plt.xlabel('Number of clusters')
plt.ylabel('WCSS')
plt.title('Elbow Method for Optimal k')
plt.show()
```

INTERMEDIATE

Q3: Why do we standardize features before K-Means?

A: To ensure all features contribute equally; otherwise, features with larger scales can dominate the clustering.

Q4: How do you interpret clusters in the Iris dataset context?

A: Each cluster ideally represents a species. Match clusters to actual species by majority vote and compare.

ADVANCED

Q5: When would the elbow method *not* show a clear-cut solution?

A: When WCSS decreases smoothly without a sharp elbow—often due to overlapping classes, noisy data, or poorly separated clusters.

Q6: If clusters don't map well to actual species, what could be the reason?

A: Some species might not be separable using only the selected features; clustering doesn't use labels, so species with overlapping feature ranges will be mixed.

Experiment 5: Random Forest Classifier for Car Safety Prediction

BASIC

Q1: What is a Random Forest, and why is it used?

A: It's an ensemble of decision trees that make predictions by voting. It's used because it reduces overfitting, handles high-dimensional data, and is robust to noise.

Q2: What is "bagging"?

A: Bagging (Bootstrap Aggregating) is the process of training each tree on a random subset of data (with replacement), which increases diversity and stability.

INTERMEDIATE

Q3: What feature types are present in the car safety data?

A: The data often includes categorical features (buying, maint, doors, persons, lug_boot, safety). Label encoding or one-hot encoding is used for these before training.

Code Query:

Show code for splitting the data and fitting a Random Forest.

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder

le = LabelEncoder()
X_encoded = X.apply(le.fit_transform)
X_train, X_test, y_train, y_test = train_test_split(X_encoded, y, test_size=0.2, random_state=42)
rf = RandomForestClassifier(n_estimators=100, random_state=42)
rf.fit(X_train, y_train)
```

ADVANCED

Q4: What happens if you set the number of trees very high?

A: Predictive performance may plateau beyond a certain point, but computational cost (time/memory) increases.

Q5: Your model has very high precision but low recall. What does this indicate?

A: The model is highly selective, predicting positives only when very certain, so it misses many true positives (could be undesirable for car safety applications).

Experiment 6: Reinforcement Learning with Maze Navigation

BASIC

Q1: Define an “agent” and “environment” in RL.

A: The agent is the learner/decision-maker. The environment is where the agent operates and receives feedback (rewards, state transitions).^[1]

Q2: What is the objective in the maze task?

A: To train the agent through trial-and-error to discover a policy that efficiently reaches the goal (maximizing cumulative rewards).

INTERMEDIATE

Q3: What’s the difference between value-based and policy-based RL methods?

A: Value-based learns the value of states or actions to choose a policy (e.g., Q-learning); policy-based learns the policy directly (e.g., REINFORCE).

Code Query:

Show pseudocode for Q-Learning update step.

```
# Q[state, action] = Q[state, action] + alpha * (reward + gamma * max(Q[next_state, :]) -
```

Q4: How do you visualize agent learning progress in a maze?

A: By plotting reward per episode or showing the path agent takes on the maze grid before and after learning.

ADVANCED

Q5: What is the “exploration vs exploitation” tradeoff?

A: The agent must choose between trying new, possibly better actions (exploration) versus exploiting known actions that give high reward (exploitation). Balancing these is key for good RL performance.

Q6: Why might your agent learn a suboptimal policy?

A: Inadequate exploration (too low epsilon), poor reward structure, or an unsuitable learning rate might cause this.

Experiment 7: Emotion Detection from Text (NLP Classification)

BASIC

Q1: What is the target variable in this task?

A: The emotion label—such as happy, sad, angry, etc.—detected from text (tweets, messages).

Q2: What vectorization method is commonly used?

A: TF-IDF (Term Frequency–Inverse Document Frequency) to convert text data into numerical feature vectors.

INTERMEDIATE

Q3: Why might you drop rare classes during data cleaning?

A: To ensure a balanced dataset—rare classes provide little data and can make models biased or underfit.

Q4: What does the confusion matrix show in this context?

A: Which emotions are predicted correctly and where the model confuses one emotion for another.

Code Query:

Show how to fit and evaluate a Logistic Regression model in this context.

```
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.linear_model import LogisticRegression

vectorizer = TfidfVectorizer()
X_tfidf = vectorizer.fit_transform(texts)
clf = LogisticRegression(multi_class='multinomial', max_iter=500)
clf.fit(X_tfidf, y)
print('Train accuracy:', clf.score(X_tfidf, y))
```

ADVANCED

Q5: Your model is stuck at around 34% accuracy with many classes. What could you do?

A: Try better cleaning, tune the model, use different features, adjust class weights, or try advanced models (e.g., neural networks or ensembles).

Q6: How can you handle class imbalance?

A: By supplying class weights during model fitting, upsampling rare classes, or downsampling frequent ones.

Q7: Show code to add class weights in Logistic Regression.

```
from sklearn.utils.class_weight import compute_class_weight

weights = compute_class_weight('balanced', classes=np.unique(y), y=y)
class_weight_dict = dict(enumerate(weights))
clf = LogisticRegression(class_weight=class_weight_dict, multi_class='multinomial', max_i
```

Let me know if you want extended code, detailed explanations, or want to practice answering any of these viva questions!

**

1. Computer-Laboratory-I-ML-Lab-Manual_print_ready.pdf