1. PCA on Wine Dataset

Q1. What is the goal of PCA?

→ Reduce dimensionality while preserving maximum variance.

Q2. Why must we standardize data before PCA?

→ To prevent high-magnitude features from dominating the components.

Q3. What does explained variance ratio indicate?

→ Percentage of total variance captured by each principal component.

Q4. How do you select the number of components?

→ Choose until cumulative explained variance ≥ 85–90%.

Q5. Write code to apply PCA with 2 components.

→ pca = PCA(n_components=2); X_pca = pca.fit_transform(X_scaled)

Q6. How to visualize PCA results?

→ Scatter plot of PC1 vs PC2, colored by wine type.

Q7. Should the target column be included in PCA?

→ No — PCA is applied only on feature matrix (X).

Q8. What happens if data is not centered?

→ First PC gets biased toward the mean direction.

Q9. PCA vs LDA — which is better for classification?

→ LDA — it uses class labels to maximize separability.

Q10. What does `whiten=True` do in PCA?

→ Scales components to have unit variance (uncorrelated + equal scale).

2. Uber Fare Prediction (Regression)

Q1. Name three preprocessing steps for Uber data.

→ Handle missing, extract hour/day, compute Haversine distance.

Q2. How do you detect outliers in fare?

→ Use IQR: remove beyond Q1−1.5×IQR and Q3+1.5×IQR.

Q3. How to check correlation between features?

→ df.corr() + sns.heatmap()

Q4. Difference between Linear, Ridge, and Lasso?

→ Linear: no penalty; Ridge: L2; Lasso: L1 (sparsity).

Q5. What is the role of alpha (α) in Ridge/Lasso?

→ Controls regularization — higher α → simpler model.

Q6. Which metric is best to compare models?

→ RMSE (in fare units); R^2 for explained variance.

Q7. Extract hour from pickup datetime.

→ df['hour'] = pd.to_datetime(df['pickup_datetime']).dt.hour

Q8. Why scale features before Ridge/Lasso?

→ Ensures all features contribute equally to penalty.

Q9. What does negative R^2 mean?

→ Model worse than predicting the mean.

Q10. How to save the trained model?

→ joblib.dump(model, 'uber_model.pkl')

3. SVM on Handwritten Digits

Q1. What is the objective of SVM?

→ Maximize margin between classes using a hyperplane.

Q2. Hard margin vs soft margin SVM?

→ Hard: no errors; soft: allows errors via C parameter.

Q3. What are support vectors?

→ Points closest to the decision boundary.

Q4. Why use kernel trick?

→ Maps data to higher dimension for non-linear separation.

Q5. Which kernel is best for digit images?

→ RBF — captures complex pixel patterns.

Q6. Load and prepare digits dataset.

→ digits = load_digits(); X = digits.data; y = digits.target

Q7. Why scale data before SVM?

→ Distance-based; unscaled features distort margin.

Q8. Train SVM with RBF kernel.

→ SVC(kernel='rbf', C=5, gamma='scale').fit(X_train, y_train)

Q9. How to evaluate SVM performance?

→ Accuracy, confusion matrix, classification report.

Q10. How to tune C and gamma?

→ GridSearchCV with cross-validation.

4. K-Means on Iris Dataset

Q1. What does K-Means clustering do?

→ Groups data into K clusters minimizing within-cluster variance.

Q2. How to determine optimal K?

→ Elbow method — plot inertia vs K, find elbow.

Q3. What is inertia in K-Means?

→ Sum of squared distances to cluster centroids.

Q4. Code snippet for elbow method.

→ for k in range(1,10): kmeans = KMeans(k).fit(X); inertias.append(kmeans.inertia_)

Q5. What is the ideal K for Iris dataset?

→ K=3 — matches three species.

Q6. Why scale features before K-Means?

→ Prevents larger-range features from dominating distance.

Q7. Visualize K-Means clusters in 2D.

→ Use PCA → scatter with cluster labels.

Q8. How to evaluate clustering quality?

→ Silhouette score or Adjusted Rand Index.

Q9. Major limitation of K-Means?

→ Assumes spherical clusters; fails on irregular shapes.

Q10. What is k-means++ initialization?

→ Selects initial centroids far apart for better convergence.

5. Random Forest for Car Evaluation

Q1. What is Random Forest?

→ Ensemble of decision trees using bagging + random features.

Q2. How does RF prevent overfitting?

→ Each tree on bootstrap + random subset; vote reduces variance.

Q3. How is feature importance calculated?

→ Average Gini decrease across all trees.

Q4. How to encode categorical variables in car data?

→ pd.get_dummies() or OrdinalEncoder for ordered categories.

Q5. Train Random Forest classifier.

→ rf = RandomForestClassifier(n_estimators=100); rf.fit(X_train, y_train)

Q6. Evaluate Random Forest model.

→ Accuracy, F1-score, confusion matrix.

Q7. What is OOB (Out-of-Bag) score?

→ Accuracy on samples not in bootstrap — internal validation.

Q8. Handle class imbalance in car dataset?

→ class_weight='balanced' or SMOTE.

Q9. How to tune Random Forest?

→ Grid search: n_estimators, max_depth, min_samples_split.

Q10. Can Random Forest be used for regression?

→ Yes — use RandomForestRegressor.

6. Tic-Tac-Toe with Q-Learning

Q1. What is Q-Learning?

→ Off-policy RL to learn optimal Q(s,a) → max future reward.

Q2. How to represent Tic-Tac-Toe state?

→ 3x3 tuple: 0=empty, 1=X (agent), -1=O.

Q3. Write the Q-Learning update rule.

→ $Q(s,a) \mathrel{+}= \alpha\,[r + \gamma \max(Q(s',a')) - Q(s,a)]$

Q4. Define a good reward function.

→ +1 win, -1 loss, 0 draw, -0.01 per move.

Q5. What is ε-greedy policy?

→ With prob ε: random; else: max Q-action.

Q6. How to initialize Q-table?

→ Q = {} (dictionary), default 0 for unseen (s,a).

Q7. How to handle invalid moves?

→ Return high negative reward or mask actions.

Q8. Structure of training loop.

→ Play episodes → update Q after each move → decay ε.

Q9. How to test the trained agent?

→ Play 1000 games vs random → compute win rate.

Q10. Save and load Q-table.

→ pickle.dump(Q, open('qtable.pkl','wb'))